Argus : An IoT Based Accident Detection System

A Project Report

Submitted by

RISHABH NAMBIAR, JEET PARTE, NAYANIKA SHETTY, AYUSH SINDHWANI, KARTIK PRAKASH

Under the Guidance of

PROF. KRISHNA SAMDANI

in fulfillment for the award of the degree of

Bachelor of Technology

in

COMPUTER ENGINEERING

At

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT AND ENGINEERING

APRIL 2018

DECLARATION

We, Rishabh Nambiar, Jeet Parte, Nayanika Shetty, Ayush Sindhwani, Kartik Prakash, Roll No. E004, E008, E016, E036, E054, B.Tech (Computer Engineering), VIII semester understand that plagiarism is defined as anyone or combination of the following:

1. Uncredited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.

2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)

3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who wrote what. (Source:IEEE, The institute, Dec. 2004)

4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidents of plagiarism in this body of work.

Signature of the Student:

Name: Rishabh Nambiar, Jeet Parte, Nayanika Shetty, Ayush Sindhwani, Kartik Prakash

Roll No.: E004, E008, E016, E036, E054

Place: Mumbai

Date

ACKNOWLEDGMENT

We, the students of **MPSTME** (Mumbai), are extremely grateful to our college for the confidence bestowed in us and entrusting our project entitled "Accident Detection System".

We give our sincere thanks to our project mentor, Prof. Krishna Samdani, to whom we are highly indebted for his guidance and constant supervision as well as for providing necessary information regarding the project.

We would like to thank our project co-ordinator, Prof. Shubha Puthran, for her valid inputs and constructive criticism which helped pave the path towards the completion of our work.

Our thanks and appreciations also go to each other as colleagues in developing the project and to people who have willingly helped us out with their abilities.

Finally, this project would not have been possible without the kind support and help of many individuals. We would like to extend our gratitude to all of them.

TABLE OF CONTENTS

List of Figures

List of Tables

Abbreviations

Abbreviation	Description
BLE	*Bluetooth*® Low Energy
IoT	Internet of Things
GSM	Global System for Mobile communication
GPIO	General-purpose input/output

ABSTRACT

According to the analysis of road accident data in India, 17 lives on average were claimed every hour due to road accidents in 2015. Among the vehicle categories, two-wheeler accounted for the highest share in total road accidents with a share of 28.8% of the total. In-spite of the laws and regulations, the number of accidents are increasing every year. Thus, we were motivated to come up with a system that will not only detect but also provide assistance if an accident or collision occurs. In this report, we have mainly focused on two-wheeler accidents and the proposed system is an accident detection and response system.

The purpose of this work is to twofold: firstly, to provide a brief review on the existing proposed accident detection systems and key-enabling technologies, and later, providing a schematic for an ideal accident detection system. Most of the existing systems make use of a microcontroller attached with a GSM module and an accelerometer to detect an accident and notify the appropriate authorities. The main disadvantage of these systems is the bulkiness and the use of a static threshold value to detect an accident. For the systems communicating with an Android application, the main disadvantage is that the communication is done over a local network or a cloud based communication. The proposed system, Argus instead makes use of a Bluetooth Low Energy (BLE) to communicate with the Android application. The system uses a Raspberry Pi Zero W, as it has built-in BLE and WiFi support and an MPU 6050 tri-axial accelerometer sensor. This improves upon the static threshold systems and thus, provides for a more reliable system by reducing the number of false alarms.

Keywords - Accident Detection, Internet of Things (IoT), Bluetooth Low Energy (BLE), Machine Learning.

CHAPTER I : INTRODUCTION

A Report on Road Accidents in India 2016, published by Transport Research wing under Ministry of Road Transport & Highways, Government of India, has revealed that four hundred deaths take place every day on Indian roads which translates into a loss of 17 lives on an average, every hour in our country. 1354 accidents occur every day in India which is nearly one accident every minute. [1] Two- wheelers accounted for the highest share in total road accidents (33.8%) in 2016 followed by cars, jeeps and taxis (23.6%) and other articulated vehicles (21.0%), buses(7.8%), auto-rickshaws (6.5%) and other motor vehicles (2.8%). Due to lack of protection, it is not a surprise that they also accounted for the highest proportion of persons killed (29.4%) out of the total number of persons killed in the country during the calendar year 2016. [5]

Ideally, when an incident occurs the rider should be provided with aid before it's too late to take any counteractive measures. Efficient emergency notification, fast transport of qualified medical personnel, correct diagnosis at the scene, stabilization of the patient, prompt transport to the point of treatment, quality emergency room and trauma care are the factors that are important in post-crash care. However, this is not the case in real life, there are a number of variables that contribute to such a situation. In most cases, the bystanders are hesitant to take any action. This is known as the Bystander Effect. Not having medical knowledge, the bystander might make the situation worse. Even if someone dares to take the road accident victim to the hospital, they are harassed by the police, hospital authorities and repeated visits of the court. They get interrogated unnecessarily, both by the hospital authorities and by the police. Whether it is sheer apathy that leads to inaction or fear of getting entangled in long-drawn police and court battles, the result remains that no one comes forward to the aid of the victim due to diffusion of responsibility. Sometimes there is no one around to help the rider, for instance, if the accident has occurred at night or it takes place in some remote area. Therefore, currently, the existing systems are inefficient to overcome the above-mentioned hindrances.

Our project provides the motorist or the rider with a sense of surety that his loved ones will be notified if anything goes wrong and a nearby hospital or emergency center will be informed about his/her accident giving the rider quick medical attention.

1.1 OVERVIEW OF THE PROPOSED SYSTEM

We propose a vehicle-resident accident detection and response system for two-wheelers with the goal of minimizing the delay between the time of accident and relief for the rider by instantly detecting an accident and dispatching emergency services. The system consists of the system relaying with an Android application. The application is installed on a smartphone, with a web server and *MPU 6050 tri-axial accelerometer sensor* integrated with the Raspberry Pi.

Argus combines the best aspects of existing technologies and provides the following features:

1. Accurate and Instantaneous accident detection.
2. Dispatchment of emergency services instantly.
3. Notification to family and guardians by querying their information from the internal database.
4. Collect and provide information about accident prone areas.

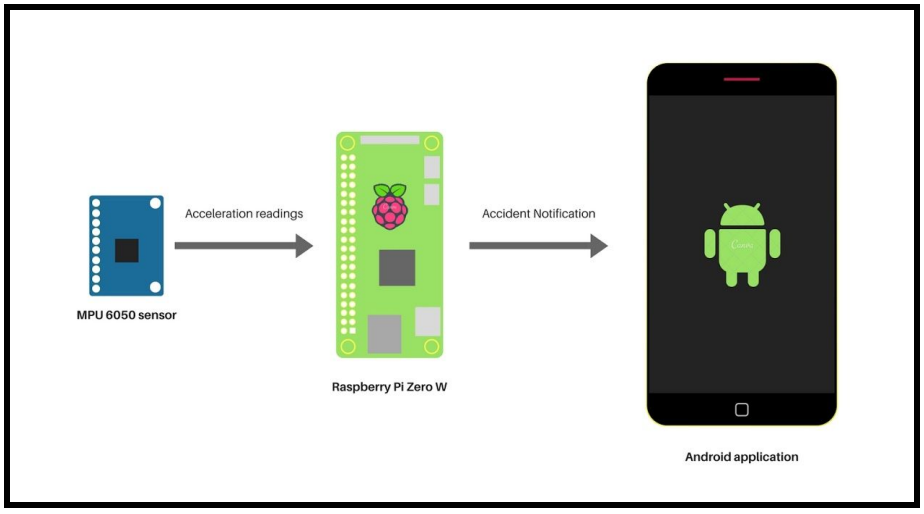

Fig 1.1 - Components and top-level overview of the workflow for the proposed system

The above diagram illustrates the components of the system and the interactions between them. The workflow and the functioning will be further elaborated in the chapter of Analysis and Design.

1.2 HARDWARE SPECIFICATIONS

1. 3-axis accelerometer(MPU-6050)
2. Raspberry Pi Zero W
3. Power supply

1.3 SOFTWARE SPECIFICATIONS

1. Mobile device running Android 4.1.1 and above.
2. JAVA for the android app development.
3. Python and/or JAVA for the accident detecting system running on the Pi Zero.

Refer to chapter of Analysis and Design for a detailed explanation to these specifications of the system

CHAPTER II : REVIEW OF LITERATURE

2.1 STUDY OF EXISTING SYSTEMS

Existing System 1 : Road Traffic Accident Detection and Ambulance Management

The system presented addresses the problem of delay minimization, right from the detection of an accident till the victim is safely handed over to the casualty. [7]

The system makes use of a SkyNav SKM53 series from Skylab M&C Co., a very high sensitivity GPS with a tracking sensitivity of -165dBm is used. It has 22 tracking/acquisition–channel receiver which enables it to get a faster initial fix even in harsh GPS visibility environments and off-road conditions. The system uses an Accelerometer MMA7260QT: a low cost triple–axis accelerometer from Freescale Semiconductor with a high sensitivity of 800mV/g is used to track the acceleration. A Raspberry Pi B+ with 512 MB of RAM keeps track of the accelerometer readings. If the reading goes above a preset threshold, the GPS coordinates are acquired over USB. An alarm is then sounded over a speaker.

The user may switch it off in case of false alarms. The dedicated in–vehicle accident detection module automatically informs the server whenever an accident happens. The design of the main server, which tracks the ambulances and dispatches the nearest ambulance to the accident spot. The android application guides the ambulance driver to the accident location.

The acceleration for a hit from free fall of the accelerometer was found from various trials to be in between 1g and 2g and for sudden brake it was found to be between 2g and 3g for a range of speeds from 10 to 80 kilometers per hour. The authors suggest an optimum threshold of 3g for crash detection.

The system provides an extensive ambulance management solution. The server is responsible for keeping track of all the ambulances, identifying the accident locations, dispatching the nearest ambulance to the accident spot and finally, monitoring the performance of the ambulance driver. A Model–View–Controller (MVC) based Django framework is used here. The POST request method requests the server to accept and store the coordinates enclosed in the body of the request. A Django app (a group of related functionalities) is used for

collecting all the accident as well as ambulance locations from POST request, applying reverse geocoding using Python's Geopy library and storing the corresponding geographical addresses in a form. The jurisdiction under which the accident has occurred is identified from the geographical address. This information on jurisdiction is later used while assigning an ambulance.

For monitoring purposes at the server, two Django views have been written, one for displaying the location of ambulances and accidents, and another one for listing the accident details in a serial manner. URLconf maps the URLs to the above mentioned views. The template here is an HTML file which uses XMLHttpRequest object for AJAX (Asynchronous Javascript and XML). XMLHttpRequest is used for asynchronous communication and it constantly monitors the concerned port for any POST requests. When an accident occurs, the server utilizes the Distance matrix service from Google Maps API web services to find the real map distances and transit times of nearby ambulances to the accident location. Only ambulances belonging to the same jurisdiction as that of the accident are considered for sorting. The ”nearest” ambulance is found by using a simple insertion sort on the transit times. The term ”nearest” is defined in terms of time required to reach the accident spot. It is to be noted that the transit time for each ambulance suggested by Google Maps API web services depends on the traffic conditions. Accident coordinates are then relayed to this ”nearest” ambulance. Utilizing other functionalities of Google Maps API web services like transit time, it is possible to track and keep a check on the performance of the ambulance driver from a single terminal preferably at a control room.

An advantage of this system is that the web application can be used on all devices (desktop/mobile) and that it comes with a comprehensive package of accident detection and ambulance management. The limitation is that the system uses hardcoded threshold values obtained from manual physical testing which is not a reliable way to classify accidents and uses a Raspberry Pi B+ which is bulkier than other Pi's and is delicate. The power source of the Raspberry Pi has not been mentioned.

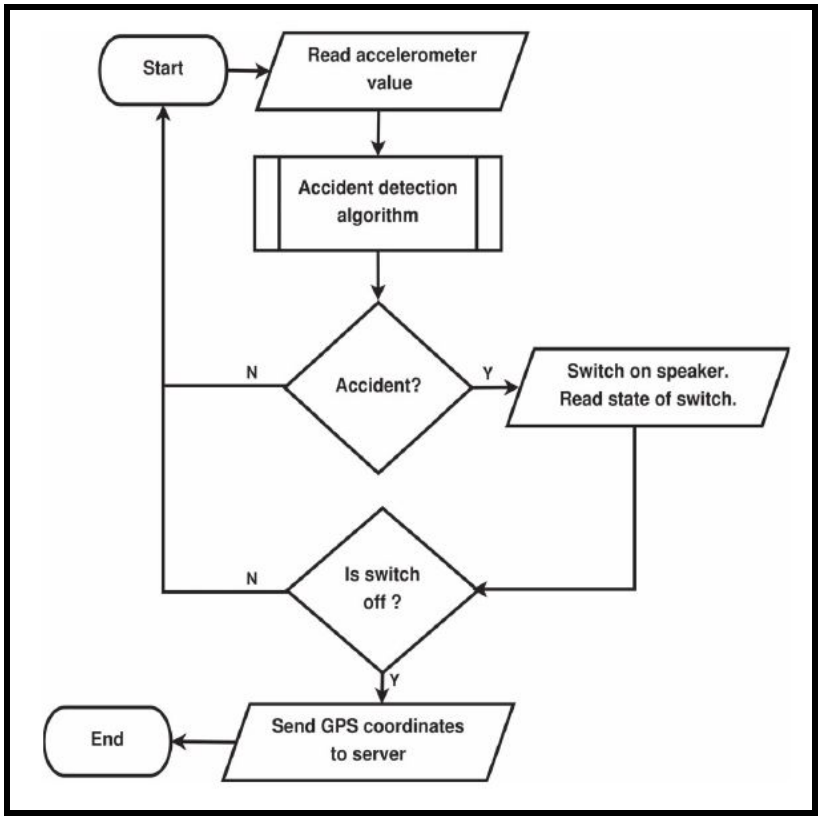

Fig 2.1 : Flowchart depicting the functioning of the system

Existing System 2 : Smart Vehicle Accident Detection and Alarming System Using a Smartphone

The system presented consists of an Android based application that that detects an accidental situation and sends emergency alert to the nearest police station and health care center. [6]

The system is implemented in five phases:

1. Bluetooth Connection, Taking Emergency Contact Number and Measurement of Parameters
2. Speed Measurement and Pressure Value
3. Change of the Tilt Angle
4. Accident Detection
5. Alarm and Emergency Alert Message

The system makes use of a pressure sensor, GPS of the mobile device and an accelerometer. The Bluetooth connection of the Android device is used to receive the pressure

sensor data. The application shows 3 flag bits initially set to ‘000’. The first bit corresponds to pressure, the second bit corresponds to the tilt and the third bit corresponds to the speed.

When the pressure sensor receives the external force that exceeds the threshold (defined as 350 in this system), the pressure bit is set to ‘1’. So, the flag number turns to ‘100’.

The tilt of the vehicle is measured using the accelerometer of the Android device. When the tilt angle makes a bigger change (greater than equal to 2 times x-axis than previous), the second bit is set to ‘2’. So, the flag number turns to ‘020’.

The application retrieves the speed value with the help of the GPS. When the speed value decreases rapidly, the third bit is set to ‘3’. It actually indicates that the current speed is less than or equal to one-third of the previous speed. So, the flag number turns to ‘003’.

In case 1, if the speed of the vehicle drops rapidly and the tilt of the vehicle is large then it considers the situation as an accident and the flag number turns to ‘023’. In case 2, when the values of speed and pressure cross the defined thresholds, it considers the situation as an accident and the flag number turns to ‘103’.

Existing System 3 : Sun SPOT based Automatic Vehicular Accident Notification System

The system presented consists of the design and development of a general purpose automatic emergency notification system for vehicles. This system, called “NOW – Notification by Wireless Systems” uses wireless channels to dispatch necessary information to emergency service providers for acquiring necessary help, with no requirement for human intervention. [8]

The figure below shows the three main components: (a) a set of wireless sensors connected to a mobile unit such as a cell phone and PDA, (b) a NOW server, and (c) a database. The sensors are planted at various strategic locations in the vehicle, with each sensor programmed to continuously collect necessary data about its environment.

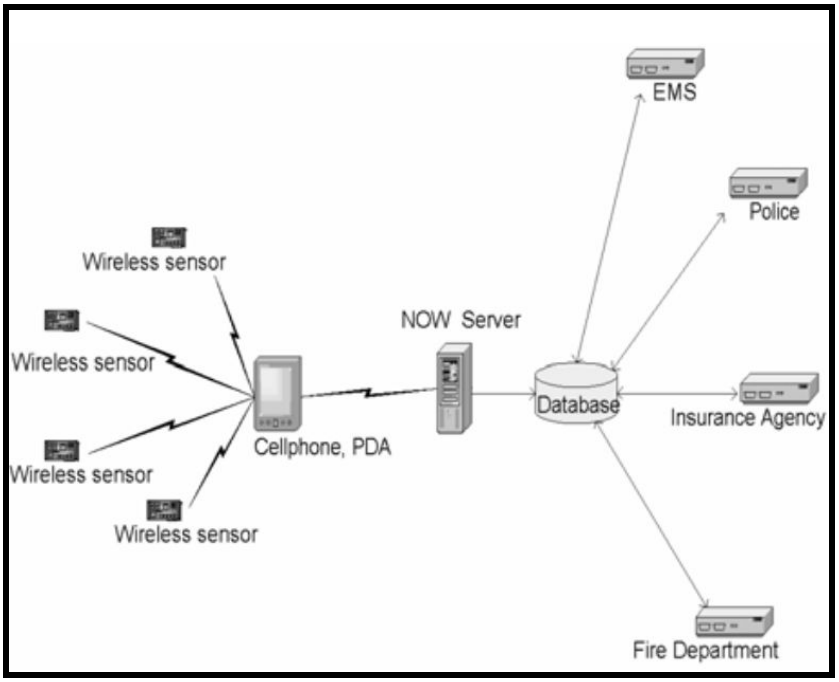

Fig 2.2 : Components of the system

The system in the paper uses Sun SPOT (Small Programmable Object Technology) and Garmin GPS10 wireless sensors to report several environmental parameters to the mobile client. The Sun SPOT hardware platform is a small, battery operated, wireless device running the Squawk Java Virtual Machine (VM) without an underlying OS. This VM acts as both operating system and software application platform. The hardware platform includes a range of built-in sensors as well as the ability to easily interface to external devices.

The task of NOW(Notification by Wireless system) mobile client is to accumulate various portions of the sensor data stream, and monitor the status of the vehicle. If an abnormality is detected, such as excessive shake, high temperature combined with high intensity light,sudden drop in speed, large angular tilt, etc., the mobile client immediately collects data, creates a specific format and sends them as a string to the NOW server for processing. NOW client does not report the EMS agencies about an accident. It is the job of the NOW server to do so.

The organization and seamless availability of various environmental data from the Sun SPOTS and GPS sensor is crucial for NOW system to work properly. This work primarily focuses on collecting data from all the sensors, creating a string containing all the data and

sending the string wirelessly to the NOW server. The accelerometer data from Sun SPOTs has a range of -90 to +90 degrees. After studying carefully the study indicates that a tilt of about 65 or more is sufficient enough for the vehicle to rollover.

Existing System 4 : An IoT Approach to Vehicle Accident Detection, Reporting, and Navigation

The system presented by conveys a smart and reliable IoT system solution which instantly notifies the PSO headquarter whenever an accident takes place and pinpoints its geographic coordinates on the map. [9]

The system is composed of the following phases:

1. Vehicle registration and preparation
2. Passengers' registration
3. Monitoring accidents through a web interface located in the PSO headquarter.

When an accident takes place, a shock sensor detects it. Then, an algorithm is applied to process the sensor signal and send the geographic location along with some ancillary information to the PSO headquarter, indicating accident occurrence. The system uses a shock sensor for detection, a Global Positioning System (GPS) - SKM53 GPS module device to send the exact vehicle location to server, Raspberry Pi open-source prototyping platform for data and signal processing, NFC Reader for reading the user's data from the mobile and Cellular IoT - cellular 3G module to establish all kind of wireless communications from and to the server.

The below system architecture is the conceptual model that defines the structure, behavior, and more views of our proposed system. It is divided into two phases :

1. Preparation/Registration Phase

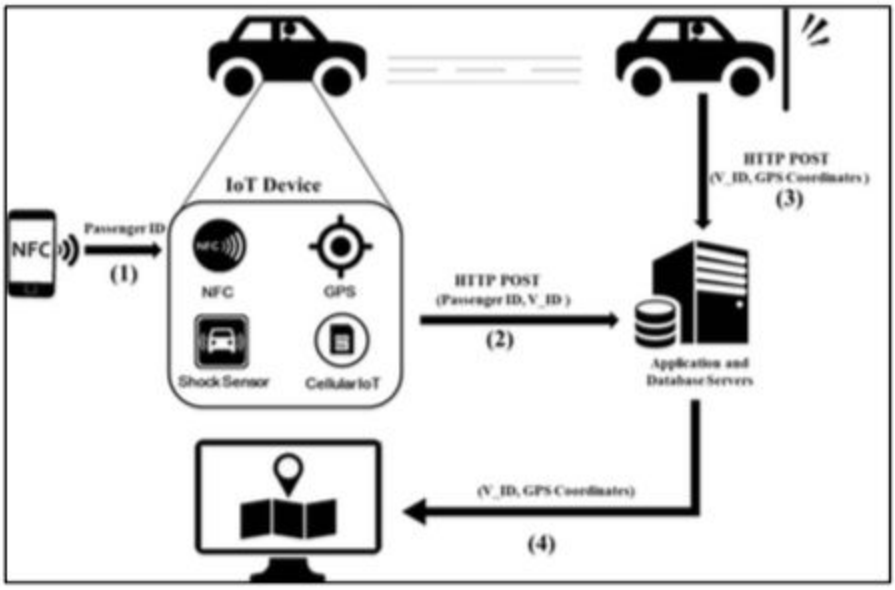

Fig 2.3 : Registration Phase Architectural Diagram

2. Monitoring Phase

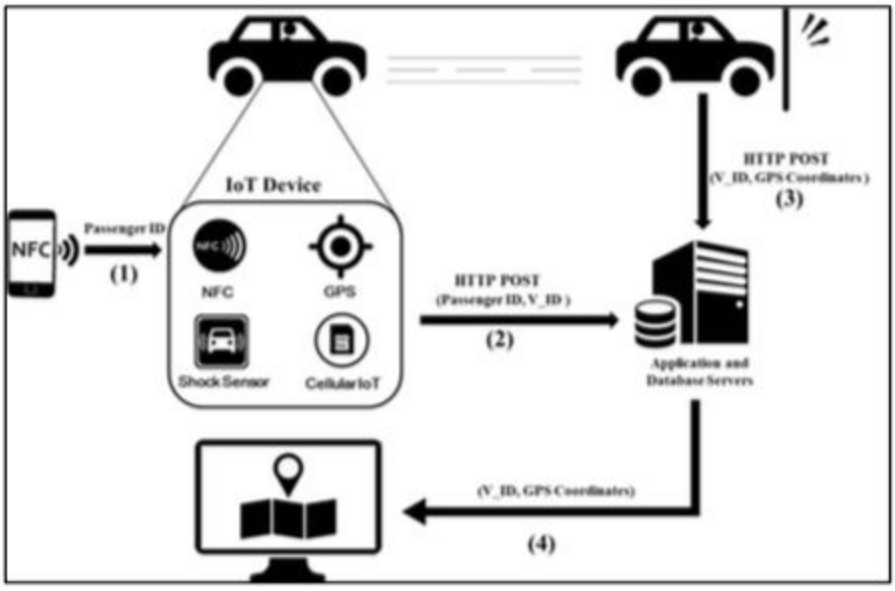

Fig 2.4 : Monitoring Phase Architectural Diagram

In registration phase, the operator registers the vehicle using its vehicle ID through a web interface connected to the database server. As a result, the Vehicle table in the database now comprises records pertaining to all registered vehicles. In monitoring phase, when the user taps the NFC enabled device (mobile phone) to the IoT node, an HTTP request holding the passenger's ID and the vehicle's ID, is sent through the IoT cellular network to the application/database servers. If a passenger decides to leave the car, he must tap again the NFC enabled device for the record to be removed from the database.

The advantages include minimizing injured passengers interaction, providing basic medical information to rescue teams, recognizing exact and accurate accidents locations, facilitating the routing process. System is robust, that is, available and serviceable and the geographical data collected from this system could be relied upon as admissible evidence or indicator of the road state and conditions. The disadvantages are that the system requires a constant internet connection, it uses only a one sensor. It does not take into account the possibility of false alarms.

Existing System 5 : Intelligent Accident-Detection And Ambulance-Response System

The system presents an intelligent accident-detection and ambulance-rescue system. [10] A sensor fitted onto the vehicle unit, along with GSM and GPS modules detects accidents and sends the accident location to a main server unit containing a database of nearby hospitals. An ambulance dispatched from one of these hospitals carries the patient to the hospital, while continuously monitoring their vitals and sending this data back to the concerned hospital. During transit, traffic signals in the path of the ambulance are controlled via RF communication, providing a clear path for the ambulance. This effectively, reduces the time for transport.

The system architecture consists of three main units that interface with a central control unit. Throughout the system, the ARM LPC2138 is the microcontroller of choice. Thus, the system consists of four units in total:

1. The Vehicle Unit
2. The Control Unit

3. The Ambulance Unit
4. The Traffic Junction Unit

The **Vehicle Unit** is installed on every vehicle. It consists of GSM and GPS modules, along with the ADXL335 accelerometer, accident detection sensors and the ARM LPC2138 microcontroller. Upon accident detection, GPS coordinates are sent to the main server (at the Control Unit) via GSM. Additionally, accelerometer data is used in a preemptive manner, forewarning the driver by playing a buzzing sound if a possibility of an accident is detected.

The **Control Unit** is the central point of the system, coordinating all the other units. It consists of a PC or dedicated mobile housing the main server and a database containing information of hospitals. Upon receiving the accident location via the GPS & GSM module of a Vehicle Unit, it consults the database to contact the nearest hospitals for ambulance dispatch.

During transit, the **Ambulance Unit** measures patient vitals - using the LM35 temperature sensor for measuring the body temperature and an IR-based obstacle sensor for obtaining the pulse rate. The data collected is then communicated to the hospital via a GSM module fitted to an ARM LPC2138 microcontroller. To clear traffic obstructions, the unit is also fitted with an RF transmitter that communicates with Traffic Junction Units installed on nearby traffic signals. **Traffic Junction Units** are fitted with RF receivers (connected to ARM LPC2138 microcontrollers) that listen for the RF transmitters on Ambulance Units. When the two units are within a 100m radius, the traffic signal corresponding to the incoming ambulance path is switched to green, thus, allowing a clear passage for the vehicle.

Since the system uses GSM (Global System for Mobile Communications), there is no distance limit or range imposed for communication between units. Also, the system integrates ambulance-rescue and traffic-control functions along with accident detection. However, it does not take into account the possibility of false alarm and so, is unable to handle them. More importantly, the accident detection algorithm is not discussed neither are any details of how the accident detection event on the Vehicle Unit is actually triggered.

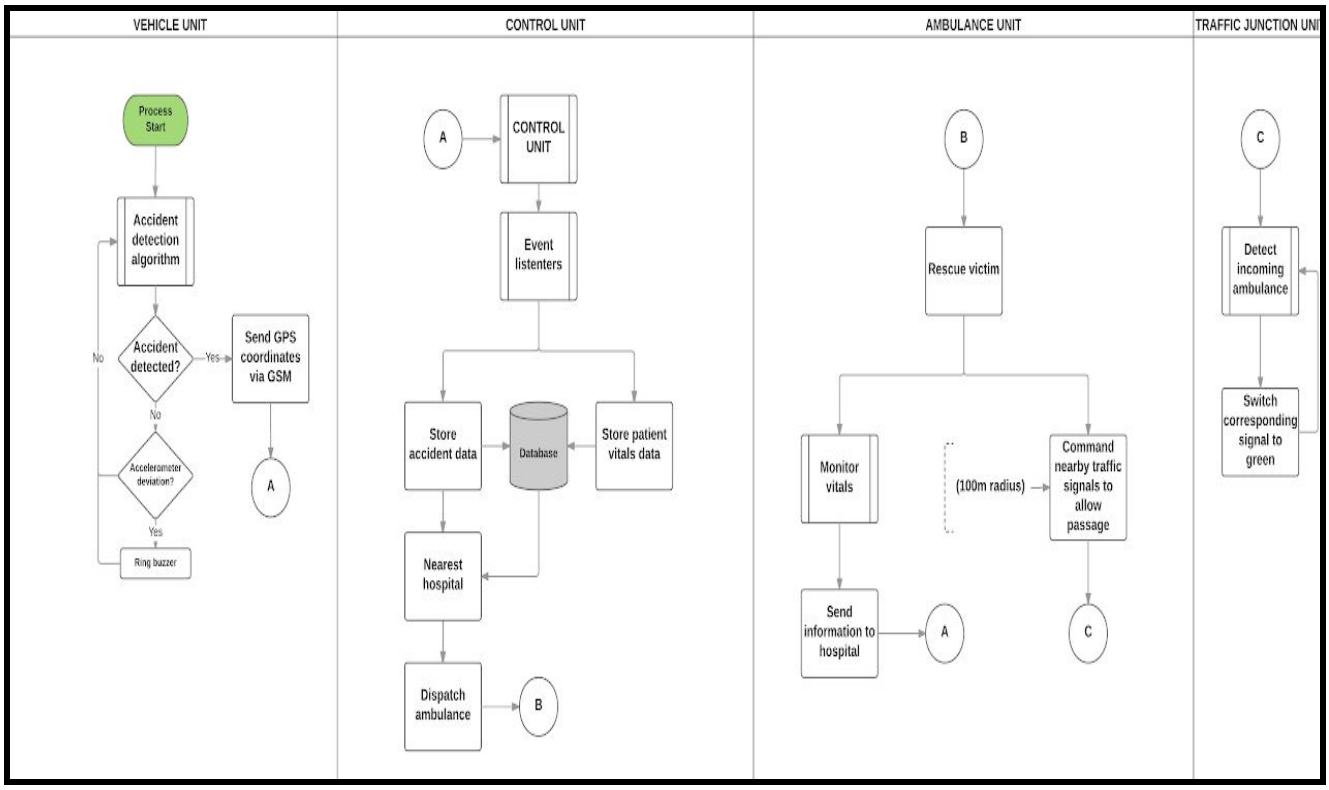

Fig 2.5 : Flowchart depicting the control flow

2.2 OUTCOME OF LITERATURE REVIEW

Majority of the existing systems make use of the GSM and GPS module integrated with the microcontroller. This not only increases the bulkiness of the system but also it makes the energy consumption of the microcontroller higher. Therefore we plan on using the GPS and GSM modules that are present in cellular devices to obtain the same results.

Another significant observation from the literature review was that the systems only used one sensor , most of the times it being an accelerometer. This limits the detection of the accident to be reliant on one type of value and therefore decreases the reliability of the system. The proposed system aims on utilising values from a triaxial accelerometer and triaxial gyroscope coupled with a crash sensor to overcome this problem. Also, in the future work the system will utilise supervised learning to further increase the reliability.

In the few systems that have made use of an android application , communication between the microcontroller and the cellular device is done using either a Wi-fi or bluetooth .

Both of the mentioned methods consume a lot of power. Therefore the introduction of BLE in the proposed system for communication helps to eradicate the problem of power consumption. Lastly, none of the papers specified as to how the microcontroller will be powered during function. The microcontroller in the proposed system will be powered by the vehicle battery.

CHAPTER III : ANALYSIS AND DESIGN

3.1 PROPOSED ARCHITECTURE OF ARGUS

Our system has a 3 tier architecture (see Fig 3.1) consisting of the following layers:

- Data Collection
- Data Processing
- Communication

Data collection: Comprises of the sensor acquiring information continuously from environment. Majorly consists of hardware components programmed to collect and store specific parameters.

Data processing: The stage where the collected data from the sensor is fed as input to the accident detection algorithm. The algorithm detects whether it is an accident based on the data collected.

Communication: Comprises of the information exchange between:

- System and the mobile application
- Mobile application and the database

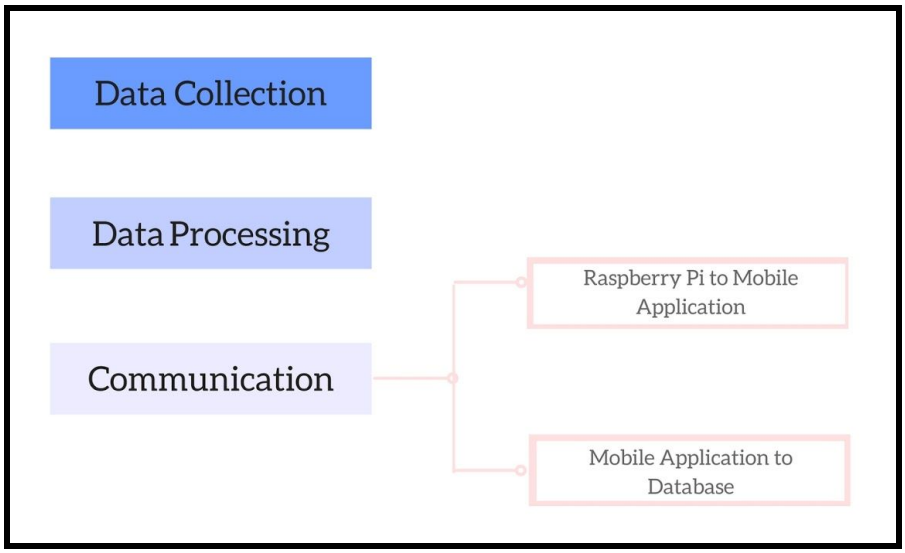

Fig 3.1 : Overview of the architecture

3.2 DESIGN AND FUNCTIONING

Figure 3.2 below is a pictorial description of the workflow of the proposed system. This figure gives us an overview on the working of Argus. The basic functioning of Argus contains 4 steps. The mentioned steps are followed when an accident occurs.

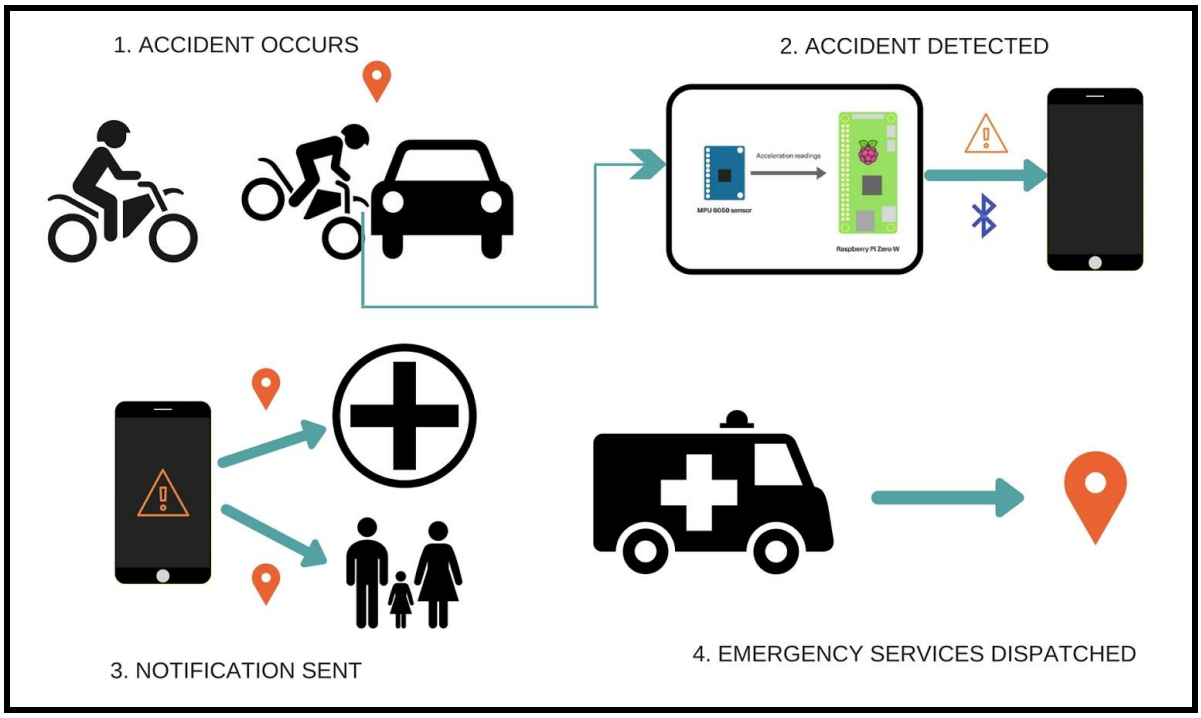

Figure 3. 2 : Workflow of the proposed system

Step 1 : When the sensor values recorded by the system pass the threshold, an accident is detected.

Step 2 : The system communicates with the Android Application on the mobile phone through BLE.

Step 3 : The Android application sends a notification to the nearest hospital requesting emergency services. This notification entails the location of the accident along with details like registered vehicle number, medical history, blood group and other details given by the user during configuration.

Step 4 : After receiving an acknowledgement from the hospital, another notification is sent to the emergency contacts added by the user at the time of configuration with the accident location and hospital details.

System - Raspberry Pi Zero W
Sensor - MPU-6050

A. Raspberry Pi Zero W

The Raspberry Pi Zero W is an incredibly cheap, incredibly small computer. At 65mm x 30mm, the Zero is smaller than a credit card, and it's only 5mm thick to boot. It also weighs just nine grams.

The Raspberry Pi Zero W extends the Pi Zero family and comes with added wireless LAN and Bluetooth connectivity. It is half the size of a Model A+, with twice the utility.

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- **Bluetooth Low Energy (BLE)**
- 1GHz, single-core CPU
- 512MB RAM
- Mini HDMI and USB On-The-Go ports
- Micro USB power
- HAT-compatible 40-pin header
- Composite video and reset headers
- CSI camera connector

The Pi Zero W is being used with a headless Linux distribution called Raspbian Stretch Lite which is a minimal image based on Debian Stretch and has the bare minimum utilities to make the data processing and communication as fast as possible.

B. MPU 6050 tri-axial accelerometer sensor

The MPU-6050™ sensors are the world's first MotionTracking devices designed for the low power, low cost, and high-performance requirements of smartphones, tablets and wearable sensors.

The MPU-6050 devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard Digital Motion Processor™ (DMP™), which processes complex 6-axis MotionFusion algorithms. The device can access external magnetometers or other sensors through an auxiliary master I²C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor. The MPU-6050 is being used over other similar sensors for facilitating straightforward I2C connections with a Raspberry Pi.

C. Bluetooth Low Energy (BLE)

Table 3.1 shows a comparison among various wireless technologies. [11] In contrast to Classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. This allows Android apps to communicate with BLE devices that have stricter power requirements, such as proximity sensors, heart rate monitors, and fitness devices. Android 4.3 (API level 18) introduces built-in platform support for Bluetooth Low Energy (BLE) in the central role and provides APIs that apps can use to discover devices, query for services, and transmit information. [12]

Common use cases include the following:

- Transferring small amounts of data between nearby devices.
- Interacting with proximity sensors like Google Beacons to give users a customized experience based on their current location.

Table 3.1 - A comparison among different low-power wireless technologies

Low Power Technology	Standard	Data Rate	Transmission Range (in m)	Target Lifetime
Bluetooth	IEEE 802.15.1	1-3 Mbs	10-50	Days- Months
Wi-Fi	802.11 b/g	11 Mbs	100	Hours
ZigBee	IEEE 802.15.4	20-250 Kbs	100	6 months-2 years
BLE	IEEE 802.15.1(V4)	1 Mbs	10	1-2 years

D. Client-Server Architecture using BLE

Using BLE, a GATT client-server communication model is setup between the Pi and the Android application. The GATT profile is a general specification for sending and receiving short pieces of data known as "attributes" over a BLE link. All current Low Energy application profiles are based on GATT. The profile describes a use case, roles and general behaviors based on the GATT functionality. Services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device. This also includes hierarchy of services, characteristics and attributes used in the attribute server. GATT is built on top of the Attribute Protocol (ATT), which uses GATT data to define the way that two Bluetooth Low Energy devices send and receive standard messages.

GATT defines client and server roles. GATT procedures can be considered to be split into three basic types: Discovery procedures, Client-initiated procedures and Server-initiated procedures. The GATT server stores the data transported over the ATT and accepts ATT requests, commands and confirmations from the GATT client. The GATT server sends responses to requests and sends indications and notifications asynchronously to the GATT client when specified events occur on the GATT server. GATT also specifies the format of data contained on the GATT server. [13]

3.3 USE CASE DIAGRAM

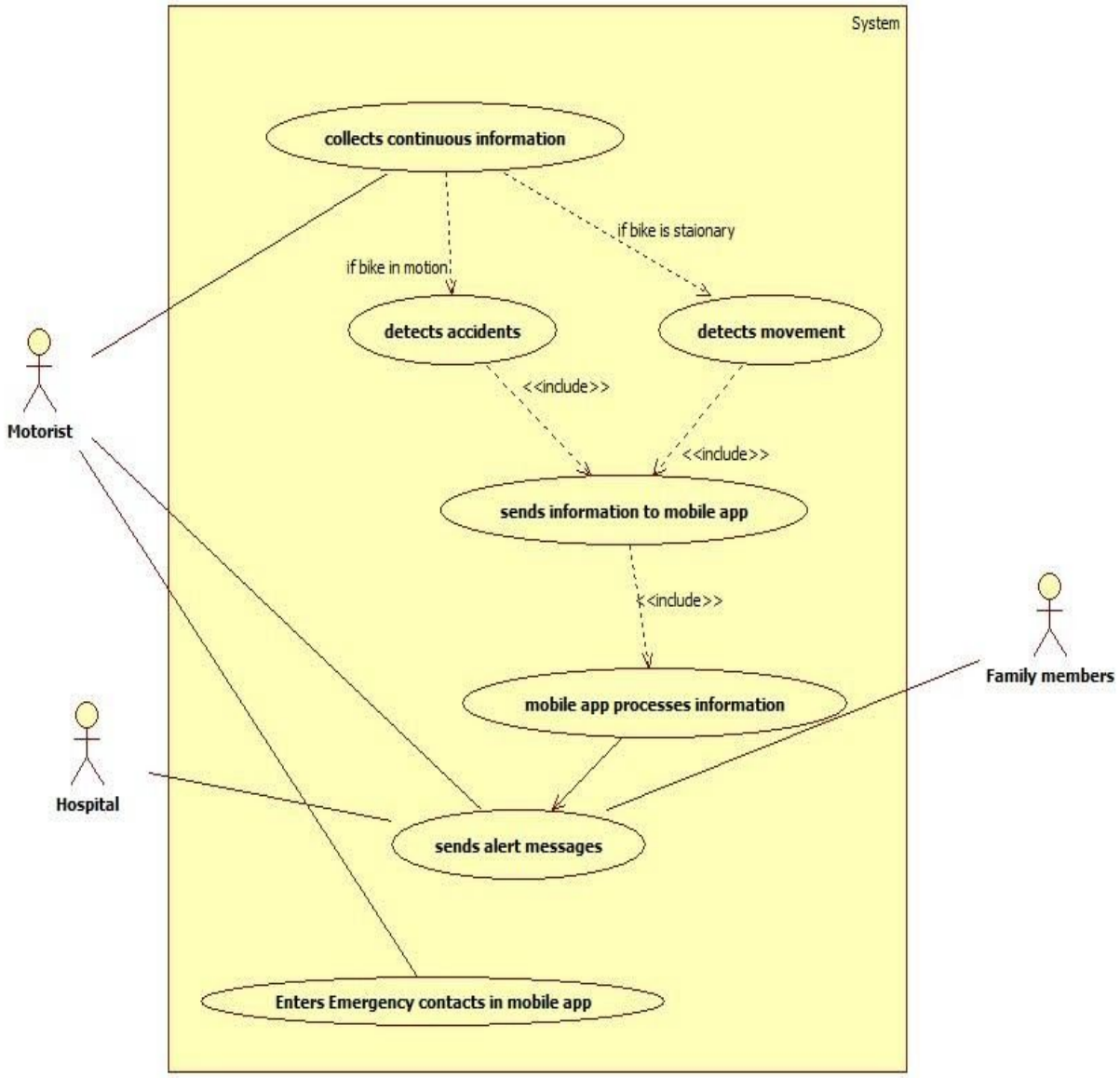

Fig 3.2 - Use case diagram for proposed system

Actors: Motorist,Family Members, Hospital

System: IoT based accident detection and notification system

Use Cases:

- Collects continuous information- Includes sensors collecting continuous values of acceleration and tilt.
- Detects accidents- Python code running on raspberry pi to detect accident from sensor data.
- Sends information to mobile application- Communication via BLE to android app to indicate occurrence of accident .
- Mobile app processes information- Prepares list of nearby emergency services and respective emergency contacts.
- Sends alert messages
- Enters emergency contacts in mobile- User enters family and relatives contacts.

3.4 FLOWCHART

The figure 3.3(a) and the figure 3.3(b) are the flowcharts related to our system. This will help understand the workflow and functioning better. Fig 3.3(a) is the flowchart overview depicting configuration setup. These steps should be followed by a user to activate the system. Fig 3.3(b) is the flowchart overview depicting the workflow of the system. This diagram takes us through the process of detection and notification which is done by the system when an accident occurs.

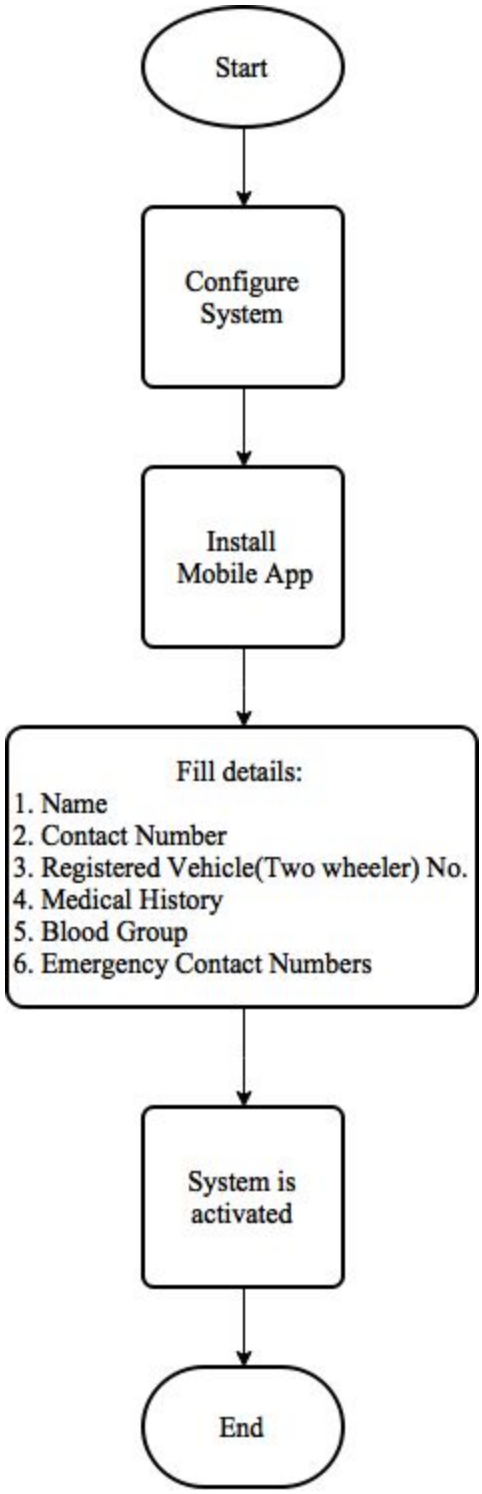

Fig 3.3(a) - Flowchart overview depicting configuration setup

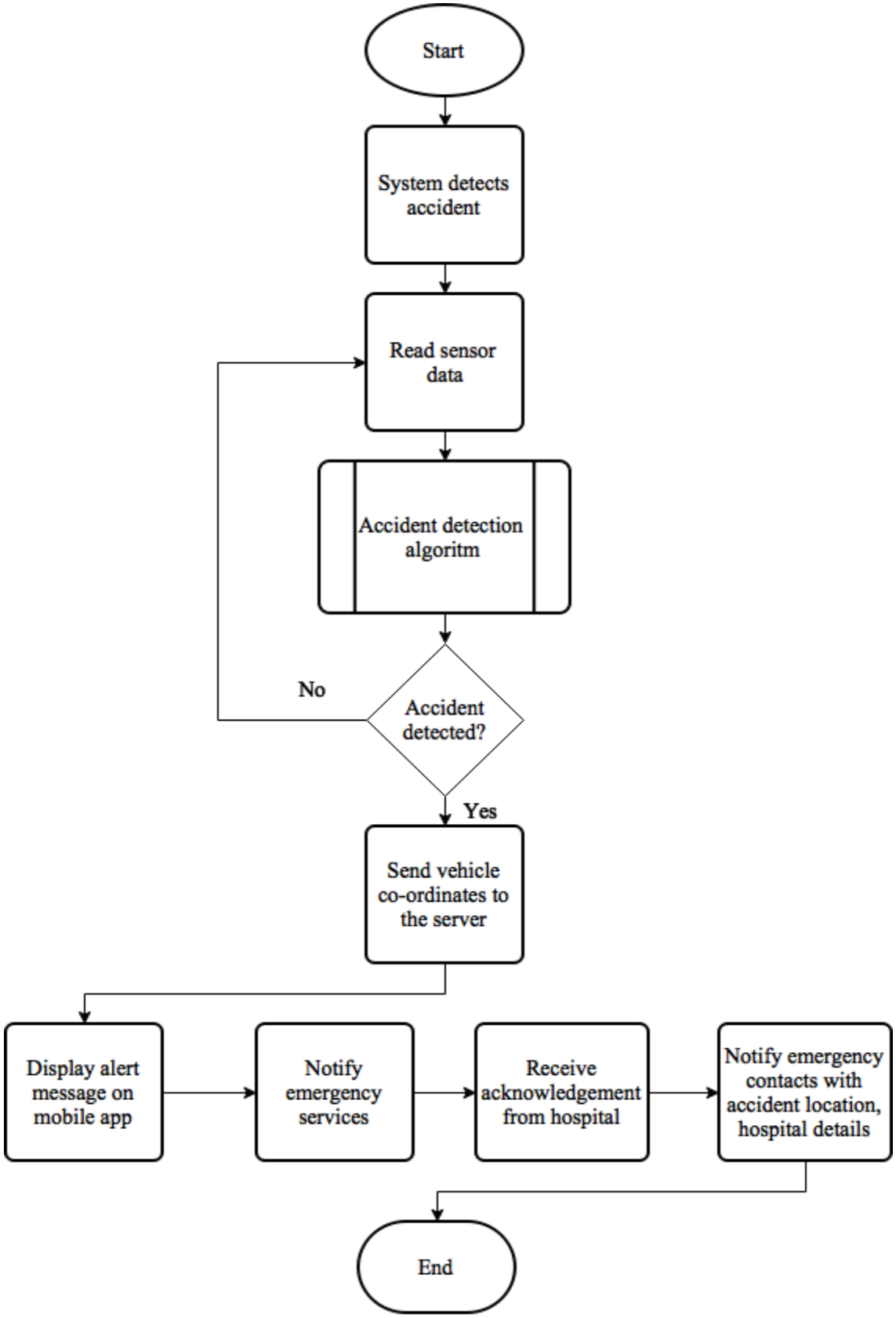

Fig 3.3(b) - Flowchart overview depicting the workflow of the system

3.5 ADDITIONAL FUNCTIONALITIES

1. False Alarm Switch (User Confirmation Switch)

Accident detection systems monitor a network of sensors to determine if an accident has occurred. Instances of high acceleration/deceleration are due to a large change in velocity over a very short period of time. These speeds are hard to attain if a vehicle is not controlled by a human driver, which simplifies accident detection since we can assume any instance of high acceleration constitutes a collision involving human drivers. Since the system contacts emergency responders—and may dispatch police/rescue teams—it is essential to identify and suppress false alarms. The inability to accurately identify and ignore false positives could render our accident detection system useless by wasting emergency responder resources responding to incident reports that were not car accidents. For such situations, we have included a false alarm switch which gives the rider a buffer time of 5 minutes after the accident has been detected. In the stipulated time, the rider can press the switch to refuse the emergency services dispatched. This ensures that the rider is safe and the event was a false alarm.

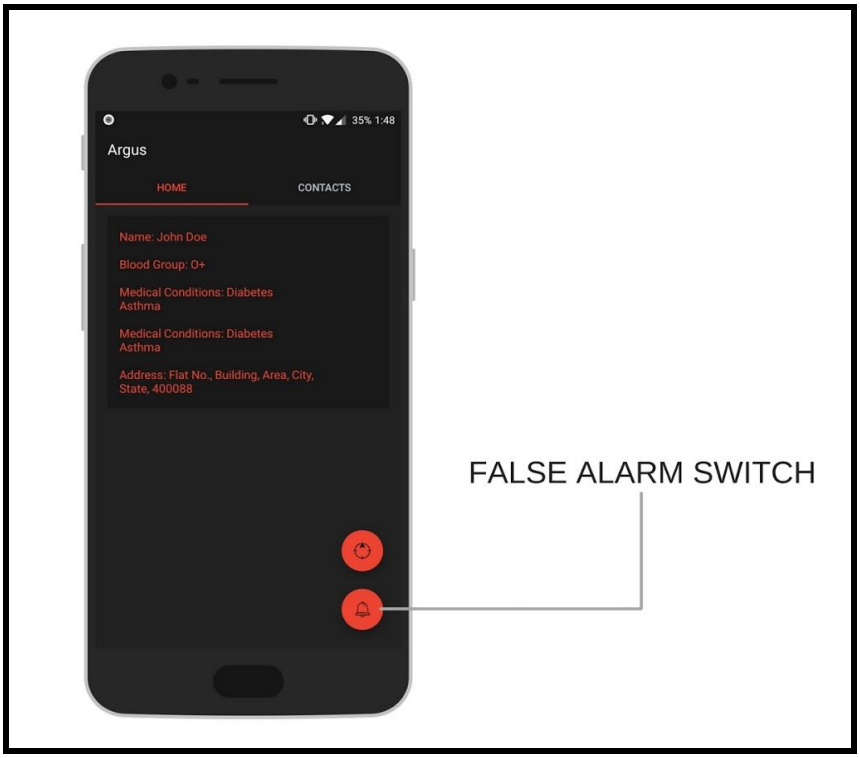

Fig 3.4 : False Alarm Switch

2. Calibration Button

The MPU-6050 needs to be calibrated before it is used in order to achieve the best possible accuracy. In the first time installation process, the rider has to place his two wheeler straight,in a neutral position, tap the calibrate button on the mobile app. In the next 10 seconds, the module will adjust for it's position and orientation on the vehicle. Calibration is a one time process; can be performed only once after installation. Refer to chapter of Methods Implemented for detailed explanation.

3.6 ANDROID APPLICATION

Given below are screenshots of the various functionalities of the android application.
Fig 3.5(a) - 3.5(j) are included in the configuration set up. Some of them include permissions.
Fig 3.5(k) is a screenshot of the message received by the emergency contact when an accident occurs.

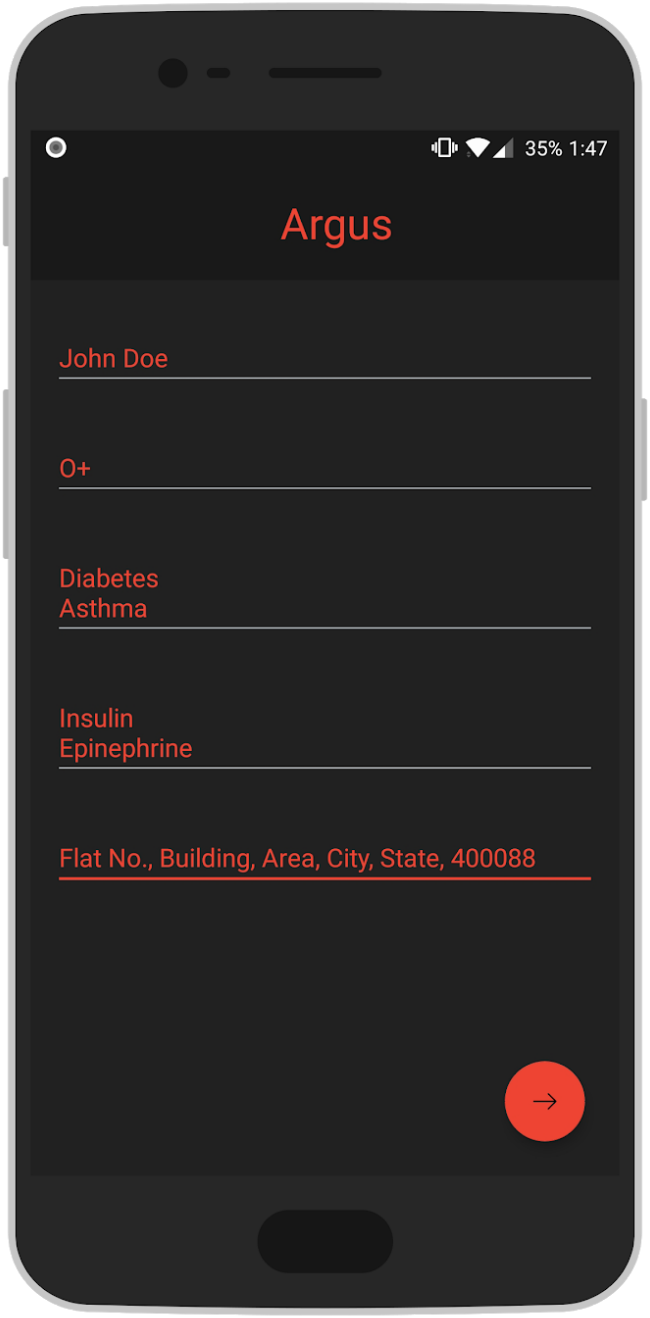

Fig 3.5(a) : Acquiring user details

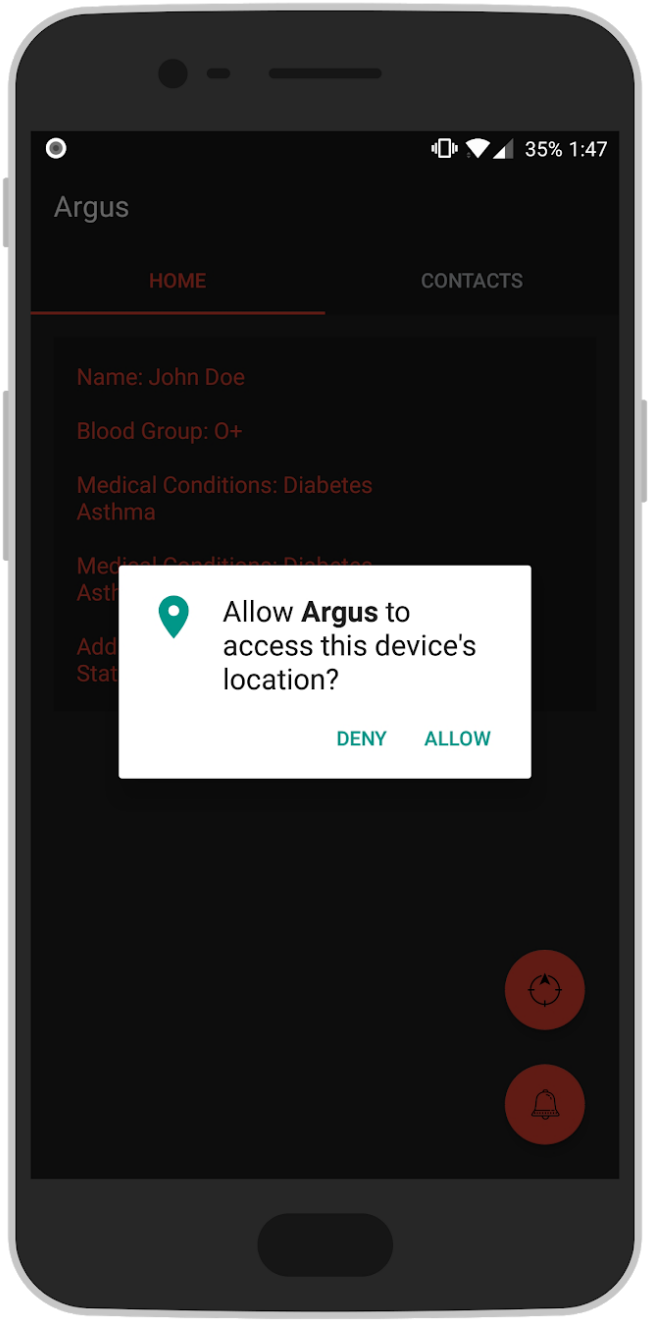

Fig 3.5(b) : Seeking permission to access the device’s location

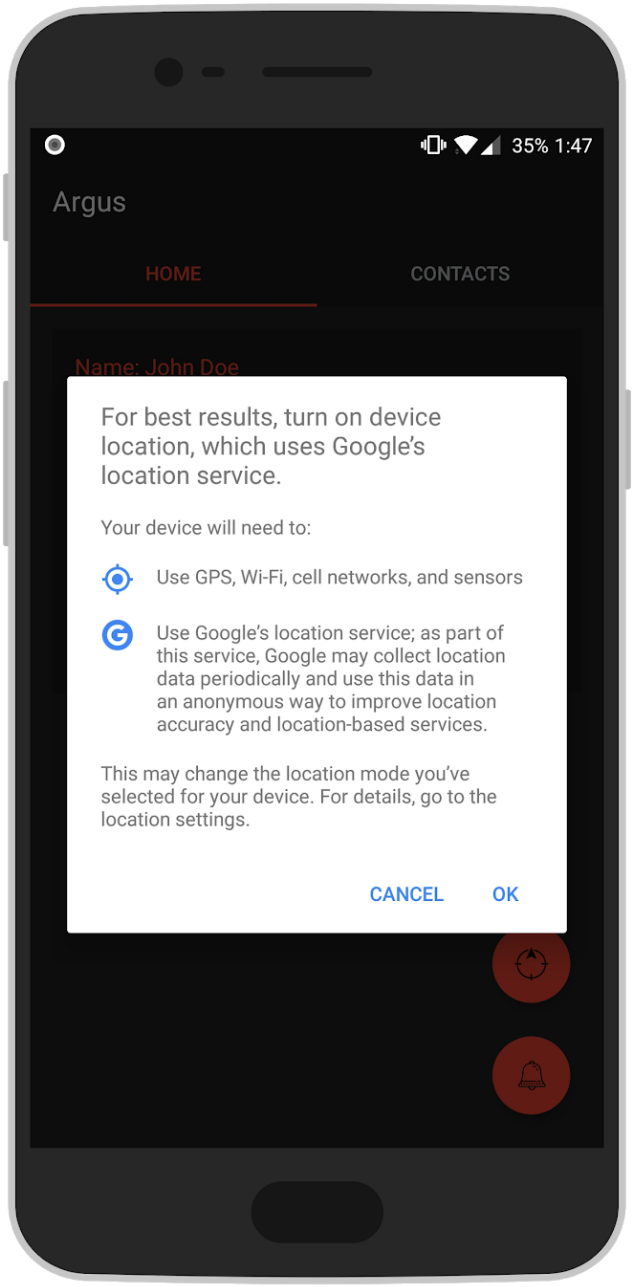

Fig 3.5(c) : Seeking permission to access the device's location

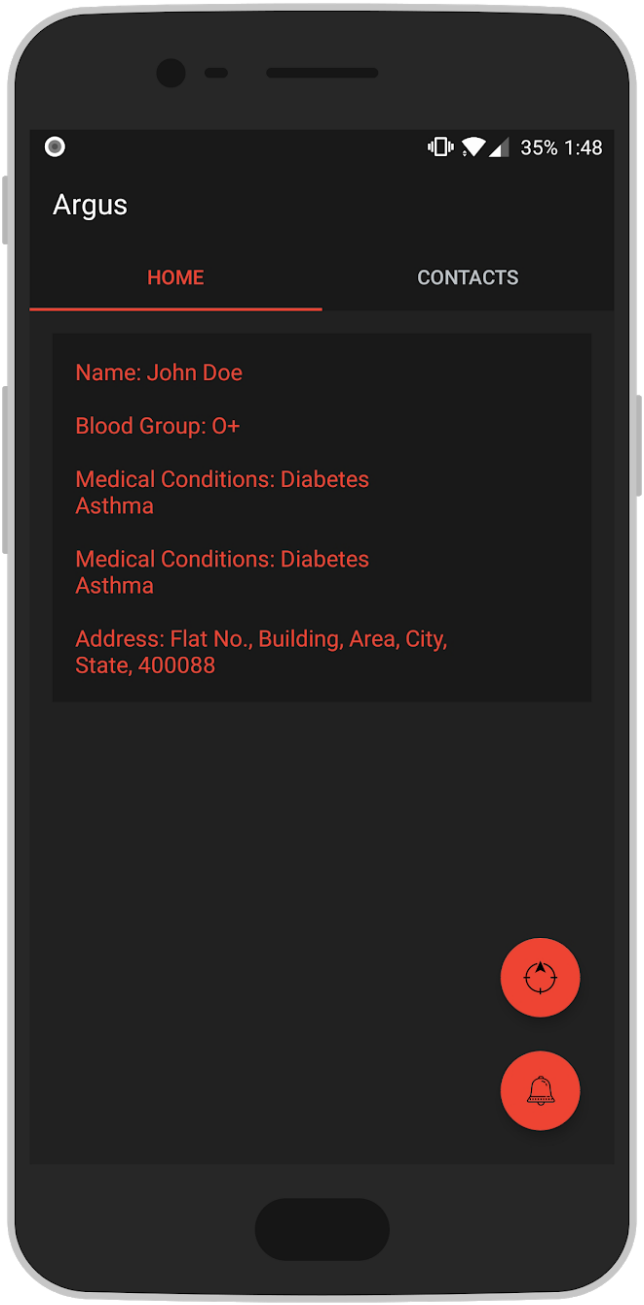

Fig 3.5(d) : Home page showing recorded details of the user

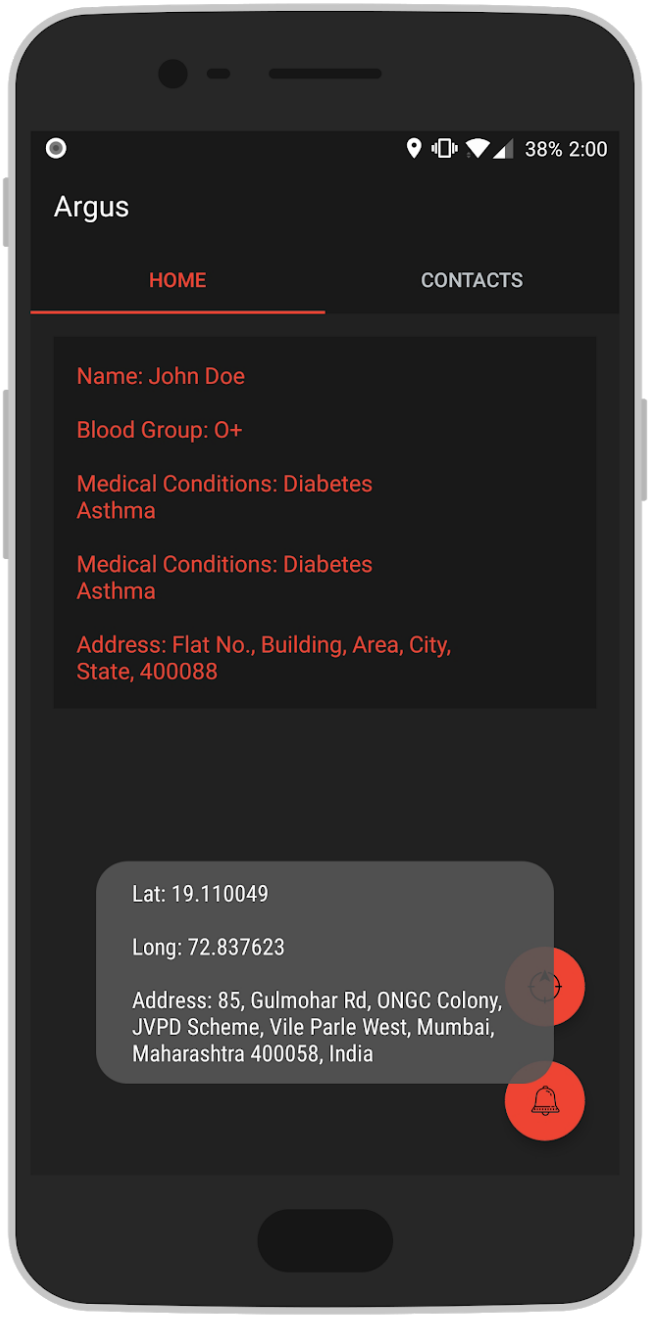

Fig 3.5(e): Home page showing recorded details of the user along with the detected location

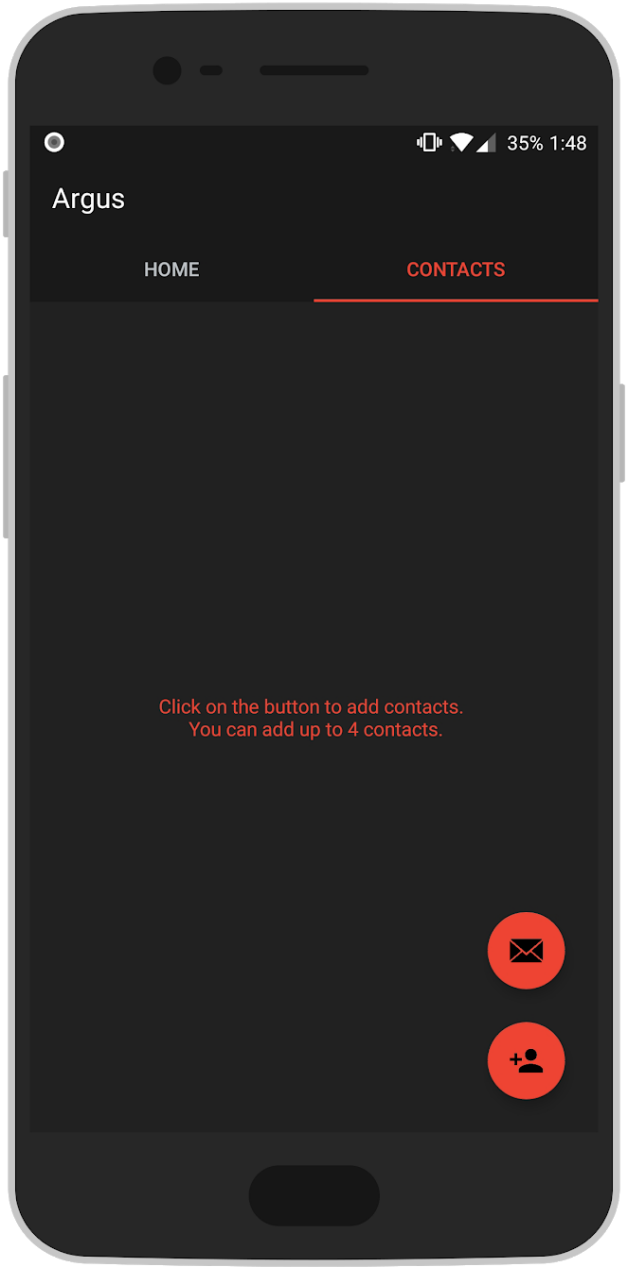

Fig 3.5(f) : ‘Add Contact’ button on the app

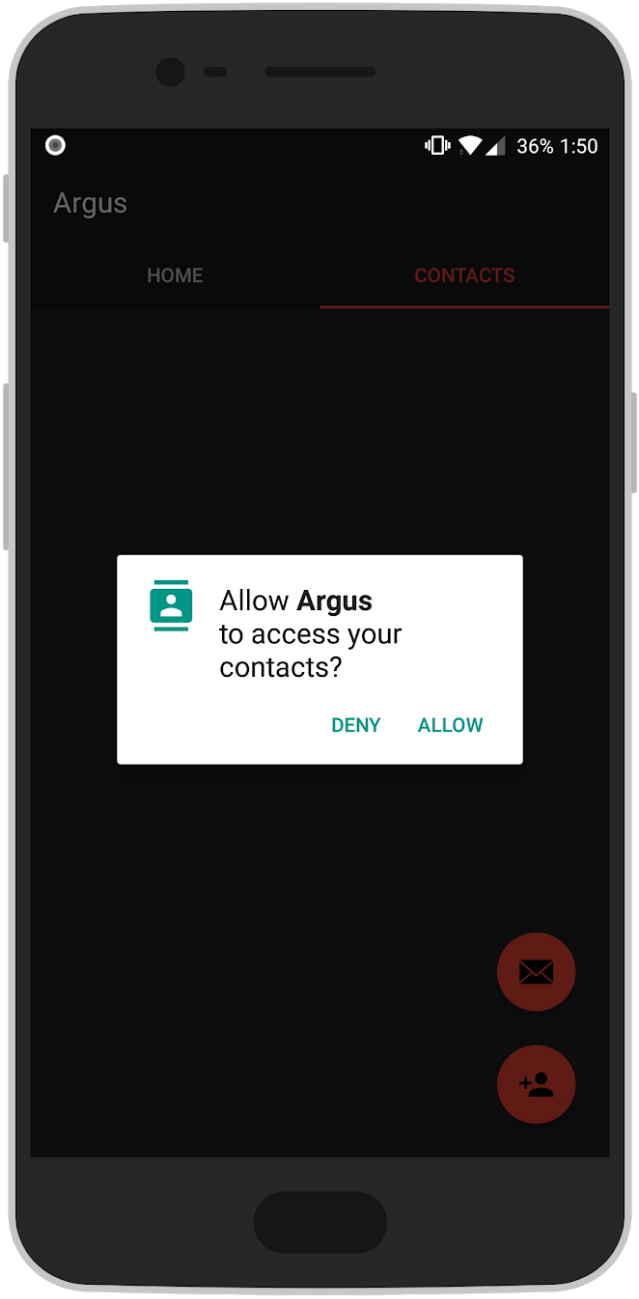

Fig 3.5(g) : Seeking permission to access the user's contacts

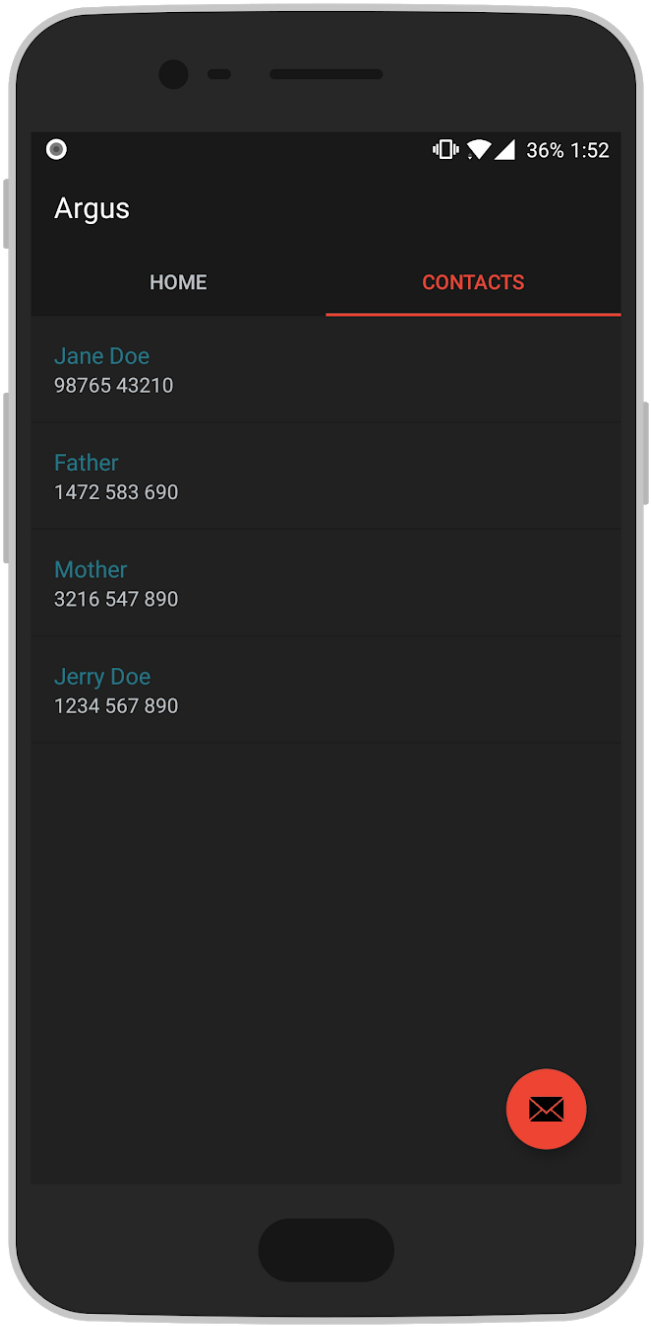

Fig 3.5(h) : List of emergency contacts added

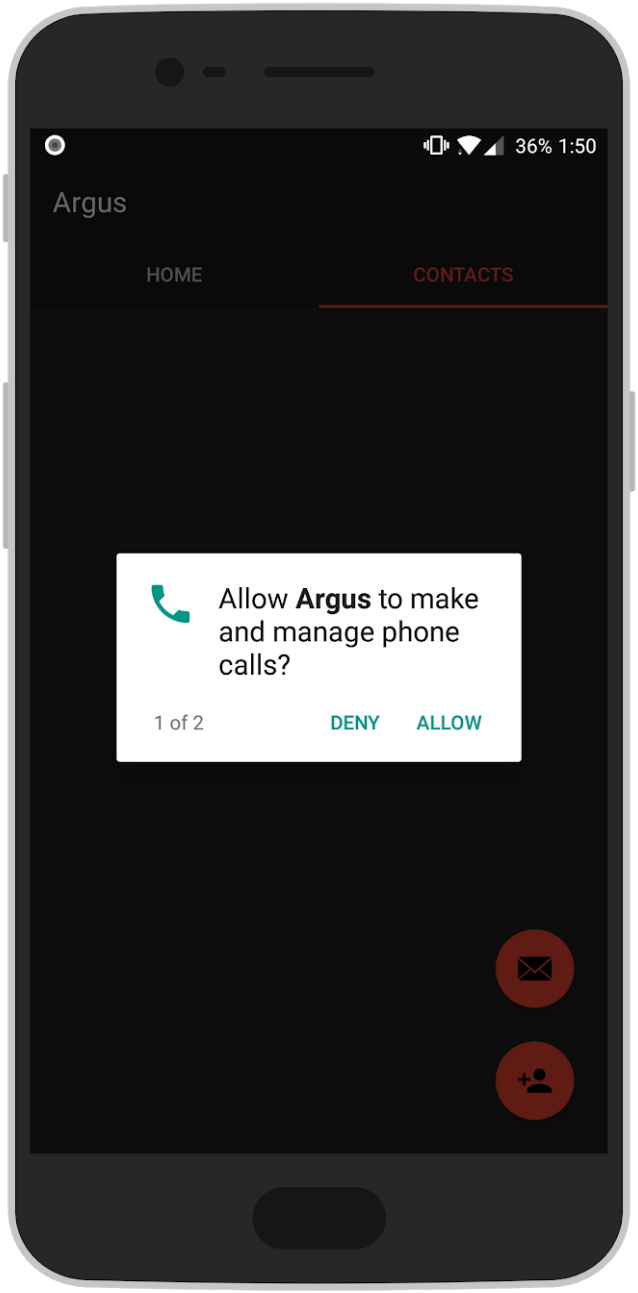

Fig 3.5(i) : Seeking permission to make phone calls

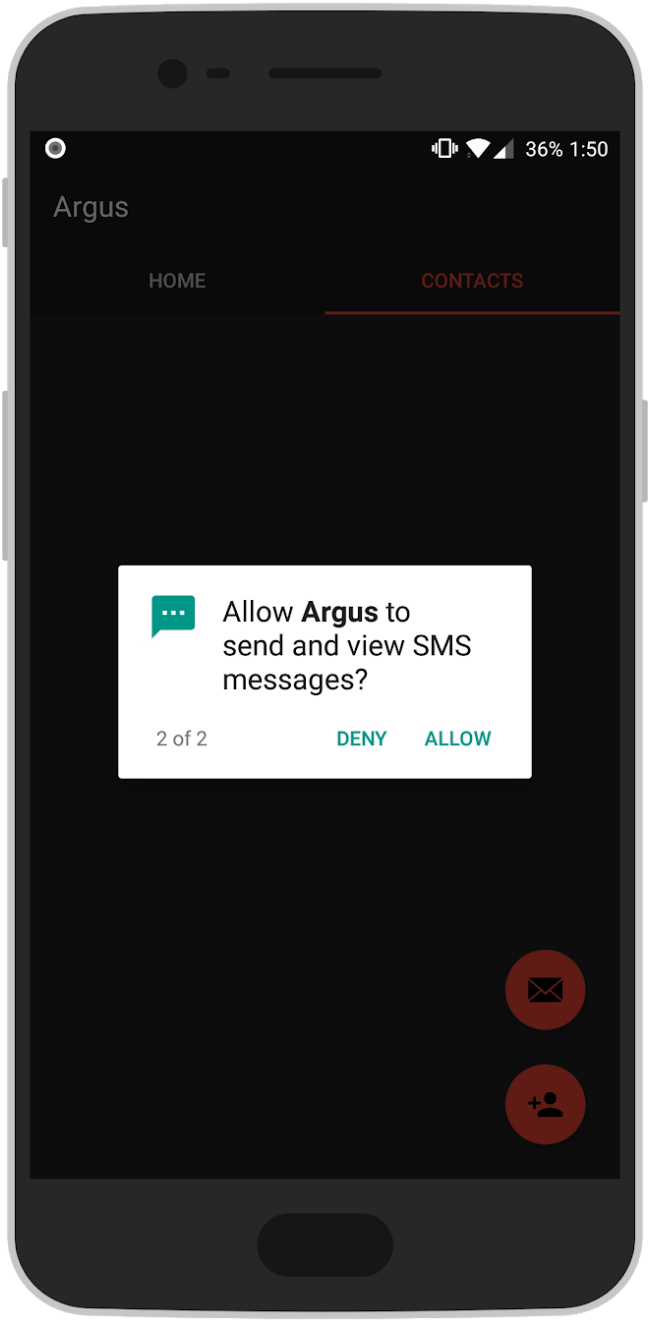

Fig 3.5(j) : Seeking permission to send messages

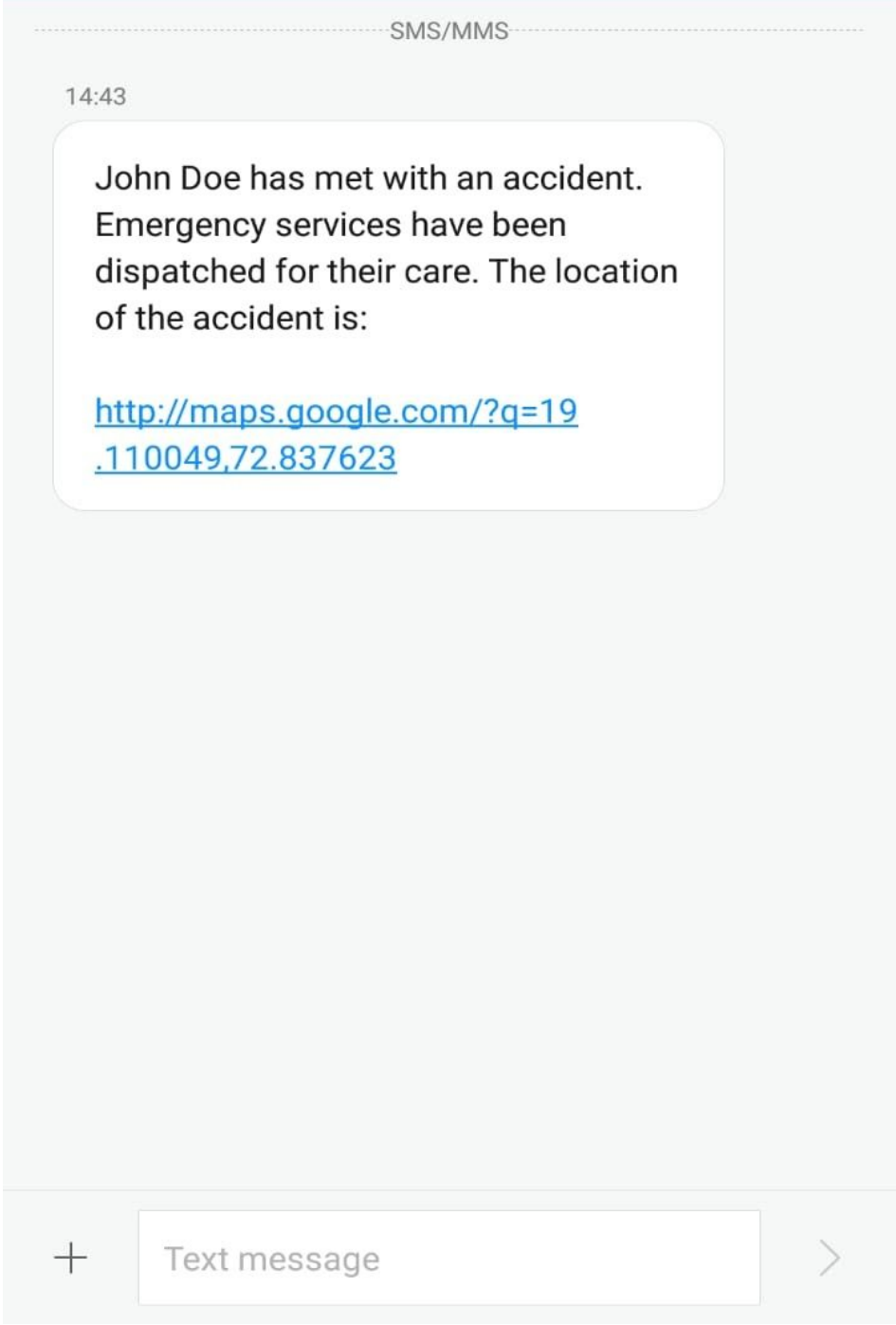

Fig 3.5(k) : Message received by emergency contact

CHAPTER IV : METHODS IMPLEMENTED

4.1 IMPLEMENTATION PROCESS

Interfacing: Pi and the sensor

- I2C -The interface between the raspberry pi zero w and the sensor
- I2C is a multi-device bus used to connect low-speed peripherals to computers and embedded systems. [14] The Raspberry Pi supports this interface on its GPIO header and it is a reliable connection protocol.
- Run the i2cdetect command to detect peripheral connected to the pi at the displayed bus address.

Connecting the sensor

To connect the sensor you need to use the GPIO pins on the Pi, the important pins are

Pin 1 - 3.3V connect to VCC Pin 5 - SCL connect to SCL

Pin 3 - SDA connect to SDA Pin 6 - Ground connect to GND

these need to be connected as shown in the Fig 4.1 below.

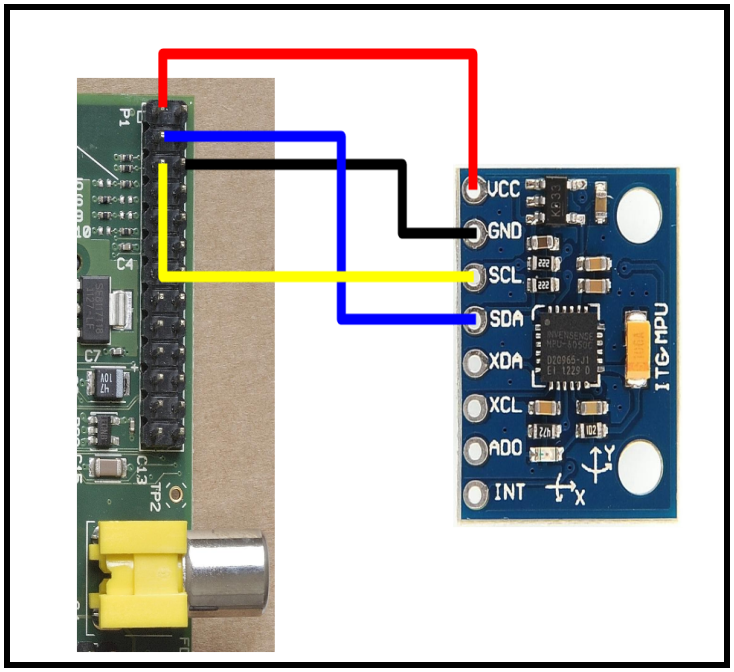

Fig 4.1 - Connections for interfacing the Pi with the MPU 6050 sensor

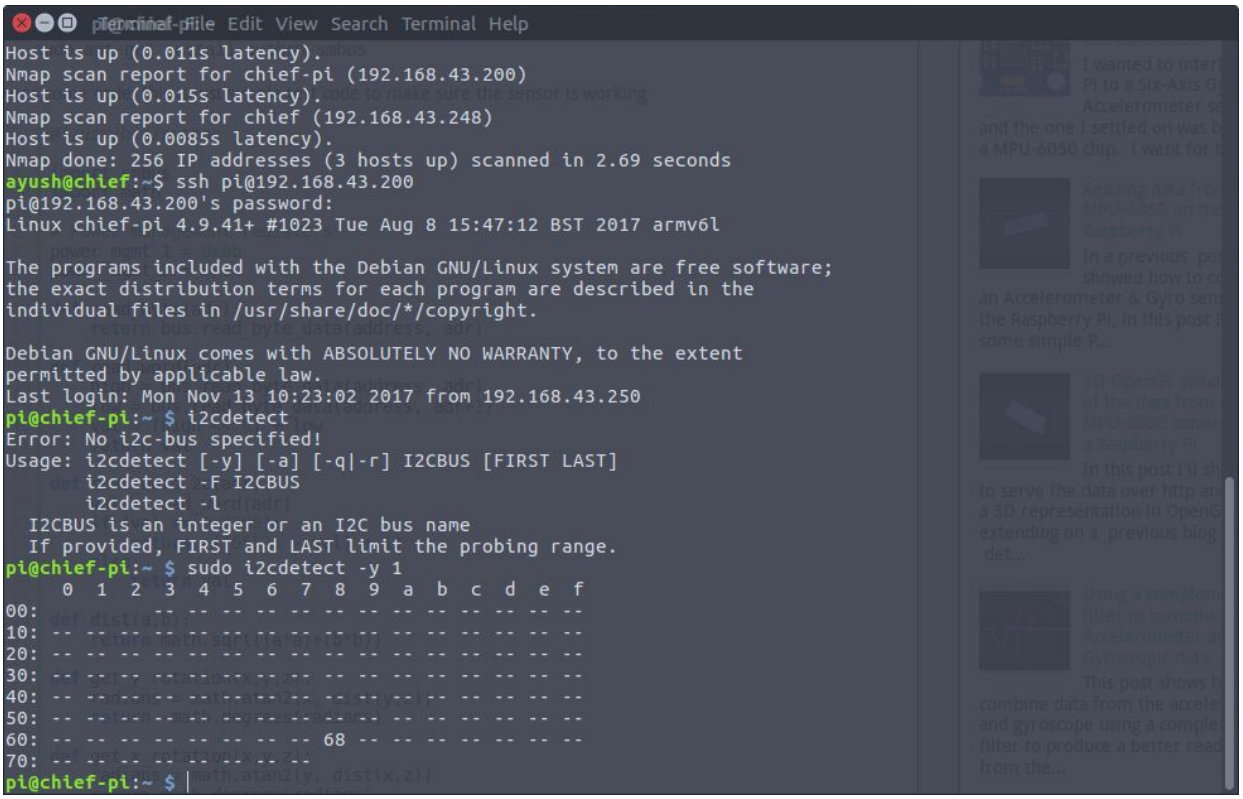

Fig 4.2 - Terminal output showing successful I2C connection between Pi and sensor

Fig 4.2 shows that the Pi has detected the sensor with an address of 0x68 (hexadecimal), this address is needed to interact with it.

Communication: The sensor and our Python code

After locating the bus address (0x68) of the sensor, we have written a Python program to utilize the features of the sensor. For communicating with the sensor, we have used a Python library called ***python-smbus.*** [15]

Python-smbus

The Python-smbus library is a Python implementation of a System Management Bus protocol. SMBus is a subset from the I2C protocol can be used for interaction between both SMBus adapters and I2C adapters.

Sample:

```
from smbus import SMBus
b = SMBus(1) # 1 indicates /dev/i2c-1
b.read_byte_data(0x68)

gyro_xout = read_word_2c(0x43)
gyro_yout = read_word_2c(0x45)
gyro_zout = read_word_2c(0x47)
accel_xout = read_word_2c(0x3b)
accel_yout = read_word_2c(0x3d)
accel_zout = read_word_2c(0x3f)
```

The above code sample demonstrates how the library can be used for reading data from the sensor.

- The first line is used to import all modules from the smbus library.

- The second line is used to create an SMBus Object by passing it a physical I2C address as a parameter. The creation of this Object will create a binding between variable **b** and the physical address of the I2C connector i.e. `/dev/i2c-1`.

- The third line is used to read sensor values from a particular address from the connection created with the I2C bus. We invoke the **read_byte_data()** function on the SMBus Object with the bus address as a parameter. Hence, b.read_byte_data will contain appropriate sensor values.
- The last six lines are then used to fetch isolated values like multidimensional gyroscopic and accelerometer data by invoking the **read_word_2c()** function and passing a bus address as a parameter.
- The sensor has a number of registers which have different functionality. The registers we are interested in for the accelerometer data are 0x3b, 0x3d, 0x3f and these hold the raw data in 16 bit two's complement format.

By writing code in this manner, we get the output given below:

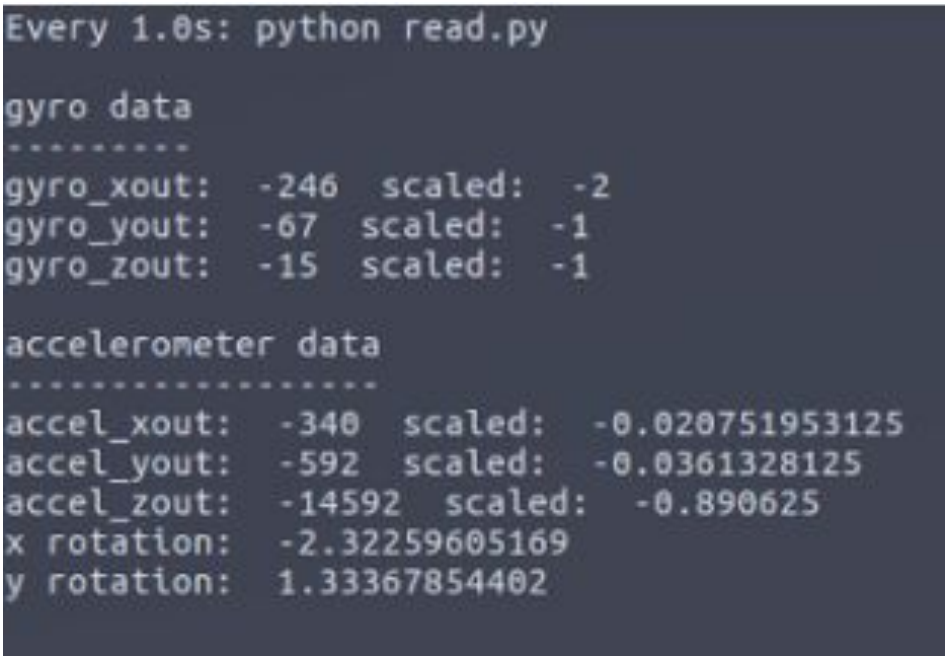

Fig 4.3 - Terminal output showing the values read into the Pi from the sensor

4.2 CALIBRATION OF MPU-6050

Sensor calibration is a method of improving sensor performance by removing structural errors in the sensor outputs. Structural errors are differences between a sensors expected output and its measured output, which show up consistently every time a new measurement is taken. Calibration provides enhanced performance by improving the overall accuracy of the underlying sensors.[16] The MPU-6050 needs to be calibrated before it is used in order to achieve the best possible accuracy. The placement of the system on the two-wheeler may vary due to the design of the two-wheeler respectively. In our case, we need to adjust the offsets in order to counteract the error caused due to situational uncertainty.

This example(Fig 4.5) is calibirate.py for calibration of the system over 10 seconds :

```
import smbus, math, time, operator, csv
from read import *

initialize()


def calibrate(seconds):
    from read import fetch_values
    gyro_sum, accel_sum = (0.0,0.0,0.0), (0.0,0.0,0.0)

    for second in range(seconds):
        seconds = (seconds, seconds, seconds)
        gyro, accel = fetch_values()
        gyro_sum = tuple(map(operator.add, gyro_sum, gyro))
        accel_sum = tuple(map(operator.add, accel_sum, accel))
        time.sleep(1)
        print gyro, accel

    gyro_avg = tuple(map(operator.div, gyro_sum, (10,10,10)))
    accel_avg = tuple(map(operator.div, accel_sum, (10,10,10)))

    x_rotation = get_x_rotation(accel_avg[0],
                                accel_avg[1],
                                accel_avg[2])

    y_rotation = get_y_rotation(accel_avg[0],
                                accel_avg[1],
                                accel_avg[2])

    rotation = (x_rotation, y_rotation)

    write_values(gyro_avg, accel_avg, rotation)

    print gyro_avg, accel_avg, rotation
    return gyro_avg, accel_avg, rotation

def write_values(g, a, r):
    with open("data.csv", "w") as file:
        csv.register_dialect("custom", delimiter=" ", skipinitialspace=True)
        writer = csv.writer(file, dialect="custom")
        writer.writerow(g)
        writer.writerow(a)
        writer.writerow(r)

calibrate(10)
```

Fig 4.4 : Snapshot of the code for calibration of MPU 6050

In the first time installation process, the rider has to place his two wheeler straight,in a neutral position, tap the calibrate button on the mobile app. In the next 10 seconds, the module will adjust for it's position and orientation on the vehicle. Calibration is a one time process; can be performed only once after installation.

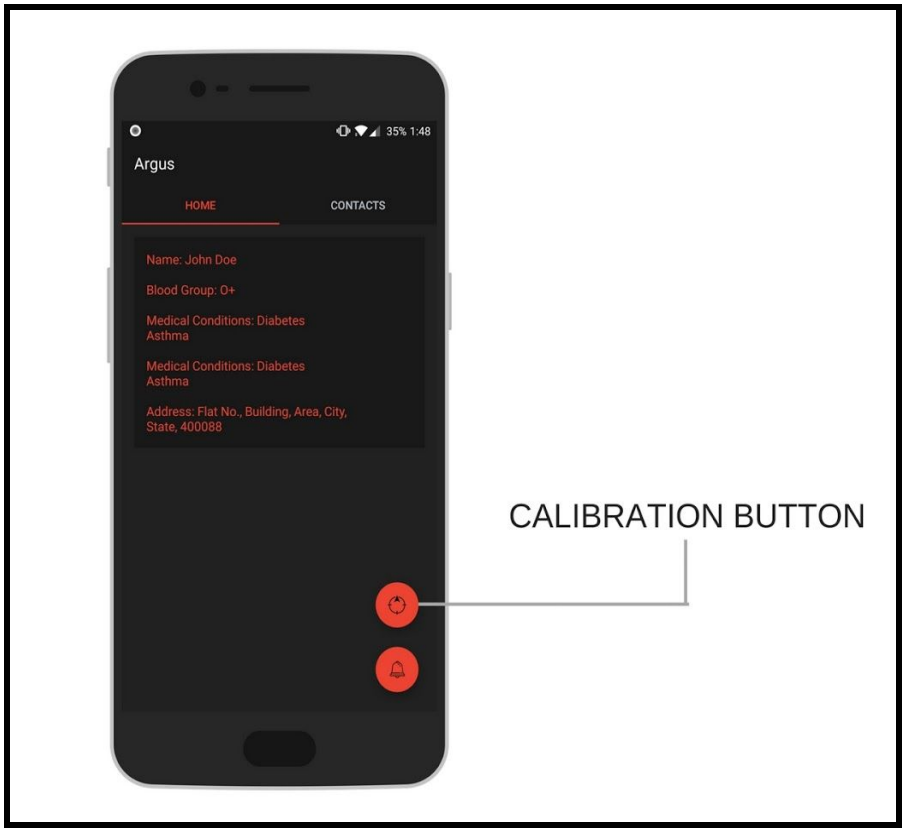

Fig 4.5 : Calibration Button

Fig 4.6 shows us the position of the calibration button on the android application. We have also created visualizations of the sensor data collected to understand the orientation of the sensor on the system. Fig 4.7a depicts the visualisation of MPU 6050 before calibration and Fig 4.7b depicts the sensor orientation after calibration.

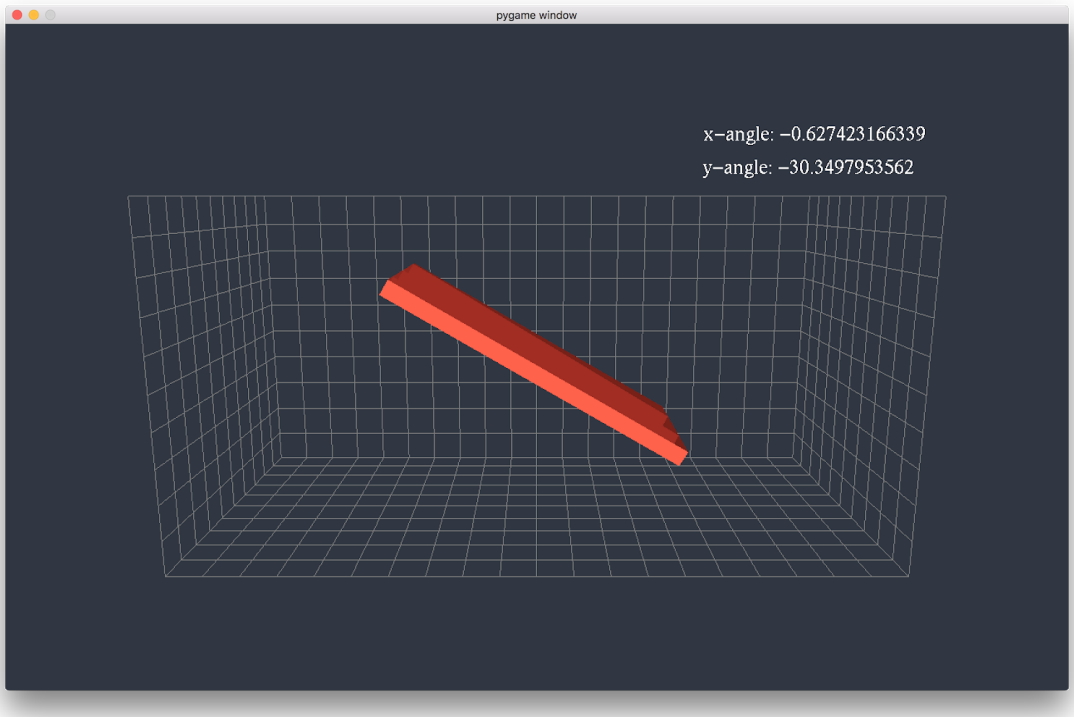

Fig 4.6a : Visualization of MPU 6050 before Calibration

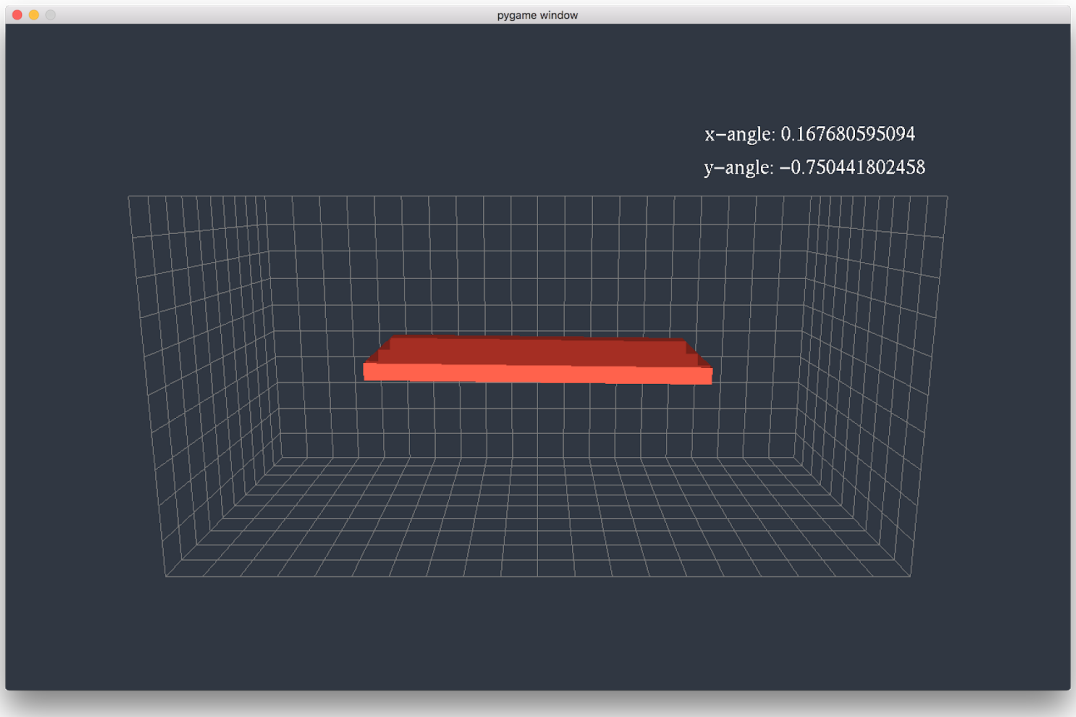

Fig 4.7b : Visualisation of MPU 6050 after calibration

4.3 DETECTION ALGORITHM

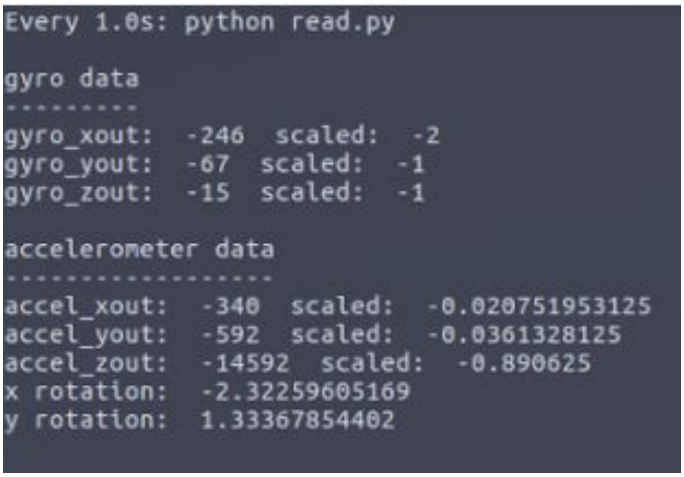

Fig 4.8. Real-time data from the MPU-6050

Using the above data, we calculate the **angular acceleration** w.r.t. the 3 axes.

The values for angular acceleration range between -256 and 256 after scaling.

If the vehicle is tilted **rapidly** beyond a certain value within a short time period , an accident is detected. Currently, accidents are detected when vehicle orientation changes **very rapidly** and when the Y-axis orientation is greater than 35 degrees or when the X-axis orientation is greater than 60 degrees.

Navigation

The Google Maps API is being used for finding the fastest route to the accident location and then to the nearest hospital. Google Maps APIs for Android are available via Google Play services so your app can be location-aware, include data-rich maps, find relevant places nearby and more. [17]

The three sub-APIs that will be used:

- Google Maps Android API
- Google Maps Geocoding API
- Google Places API for Android

CHAPTER V : RESULTS AND DISCUSSION

5.1 SENSOR VALUES

As seen in the above figure, the sensor yields two main classes of values:

1) Gyroscopic Data

A gyroscope is a device that uses Earth's gravity to help determine orientation. Its design consists of a freely-rotating disk called a rotor, mounted onto a spinning axis in the center of a larger and more stable wheel. As the axis turns, the rotor remains stationary to indicate the central gravitational pull, and thus determines orientation.

Gyroscopic information is being used with three variables:

- Gyro_**x**out
- Gyro_**y**out
- Gyro_**z**out

The **Gyro_out** values gives us the degrees per second rotation value.

2) Accelerometer Data

An accelerometer is an electromechanical device that is used to measure acceleration and the force producing it. . When the object it's integrated into goes from a standstill to any velocity, the accelerometer is designed to respond to the vibrations associated with such movement. It uses microscopic crystals that go under stress when vibrations occur, and from that stress a voltage is generated to create a reading on any acceleration.

Accelerometer information is being used with five variables:

- Accel_**x**out
- Accel_**y**out
- Accel_**z**out
- **X**_rotation
- **Y**_rotation

The **Accel_out** values provide a real time stream of the module's raw accelerometer values in X,Y and Z and the **X_rotation** or **Y_rotation** value provides rotation angle in degrees.

So, we now have the following variables that from the sensor:

- Gyro_**x**out
- Gyro_**y**out
- Gyro_**z**out
- Accel_**x**out
- Accel_**y**out
- Accel_**z**out
- **X**_rotation
- **Y**_rotation

We can query these values at any rate or set up a feed of instantaneous/real time sensor values for further processing. We could setup a webserver API that returns values when it's functions are invoked. These variables can give us insights into various scenarios such as rapid **acceleration** and **deceleration**, **tilt** of the vehicle, **twisting** motion and **disorientation** of the vehicle and in case of a **crash**.

5.2 PERFORMANCE ANALYSIS

The system was tested by fitting it on an RC car and running 50 simulations using the detection algorithm.

The car was:

- Driven through a ramp jump
- Bumped into a wall
- Flipped over
- Dropped from above
- Jerked and moved rapidly

Table 5.1 : Test Cases

Gyro y-out scaled	Angular x-rotation	Angular y-rotation	Accident Detected
104	2	-36	Yes
126	45	100	Yes
209	67	20	Yes
-254	43	29	Yes
199	23	-77	Yes
128	34	26	No
191	3	10	No

Accidents were detected in all of these cases and the threshold values were fine-tuned according to the outputs. The threshold values were decided by testing the acceleration and gyroscopic values when the module was inside a moving vehicle. The next version of the algorithm will be improved by adding more scenarios and new threshold values made for these scenarios.

5.3 COMPARISON ANALYSIS

The main disadvantages of existing systems are the bulkiness and the power consumption of the system. Majority of the existing systems make use of the GSM and GPS module integrated with the microcontroller which increases the bulkiness of the system but also makes the energy consumption of the microcontroller higher. In the few systems that have made use of an android application, communication between the microcontroller and the cellular device is done using either a Wi-fi or bluetooth . Both of the mentioned methods consume a lot of power.

Argus overcomes the shortcomings of the existing systems as follows :

1. The system utilizes the Raspberry Pi Zero W which uses less than half the power of any other Pi.
2. The Raspberry Pi Zero W has built in BLE (Bluetooth Low Energy); the system makes use of BLE to communicate with the Android application. In contrast to Classic Bluetooth, Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption.
3. The system is robust and not as bulky as compared to other systems.

The Raspberry Pi is a low-power device, which supports being powered via USB. The Table # below shows the comparison of power consumption of some models of the Raspberry Pi measured using PowerJive USB Power Meter. The Zero uses a similar amount of power as the A+, but both use less than half the power of any other Pi.

Table 5.2 : Power Consumption of Pi Models

Pi Model	Pi State	Power Consumption
A+	Idle, HDMI disabled, LED disabled	80 mA (0.4W)
A+	Idle, HDMI disabled, LED disabled, USB WiFi adapter	160 mA (0.8W)
B+	Idle, HDMI disabled, LED disabled	180 mA (0.9W)
B+	Idle, HDMI disabled, LED disabled, USB WiFi adapter	220 mA (1.1W)
model 2 B	Idle, HDMI disabled, LED disabled	200 mA (1.0W)
model 2 B	Idle, HDMI disabled, LED disabled, USB WiFi adapter	240 mA (1.2W)
Zero	Idle, HDMI disabled, LED disabled	80 mA (0.4W)
Zero	Idle, HDMI disabled, LED disabled, USB WiFi adapter	120 mA (0.7W)

The graph below explains the power consumption of various models through idling, loading up the operating system, watching video, and shooting video just for our understanding. The graph indicates the the Zero W requires about 20 mA more than the non-wireless Zero, but that's still almost half what the Pi 3 needs.[18]

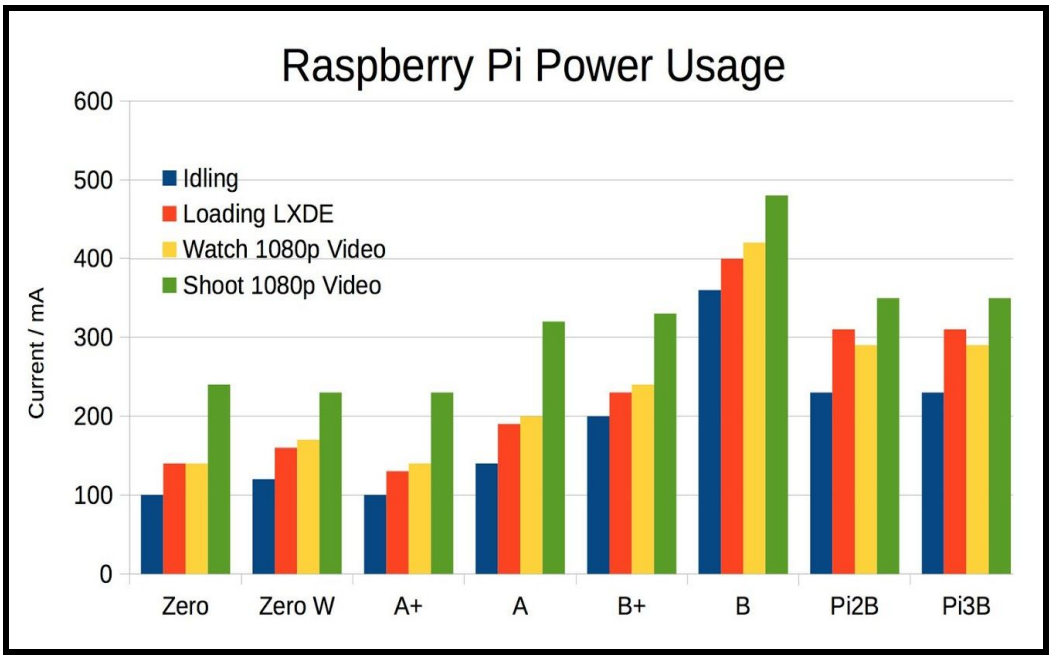

Fig 5.1 : Graph mapping power usage in Pi models

Now coming to BLE, the table below shows how BLE trumps classic bluetooth and WiFi as it's peak current draw is much lesser than the latter two.

Table 5.3 : Comparison of wireless technologies

	Classic Bluetooth	Bluetooth LE	WiFi
Standards Body	Bluetooth SIG	**Bluetooth SIG**	WiFi Alliance
Network Standard	IEEE 802.15.1	**IEEE 802.15.1**	IEEE 802.11n
Range	100m (330ft)	**50m (160ft)**	70-250m (230-820ft)
Frequency	2.4-2.5GHz	**2.4-2.5GHz**	2.4-5GHz
Over the air data rate	2.1 Mbps	**1Mbps**	Up to 600Mbps w 4x4 MIMO
App throughput	0.7-2.1Mbps	**0.27 Mbps**	Varies
Latency	100ms	**6ms**	<1ms
Peak current consumption	<30mA (Varies)	**<15mA (read and transmit)**	~50mA (Read) ~200mA (Transmit)

Some other reasons to use BLE over WiFi are :

1. Privacy

BLE offers more privacy as you have to switch on the Bluetooth facility on your phone and allow location detection. Wi-Fi technology does not necessarily ask the permission, because there is no user intervention involved. If you want to be free of Wi-Fi, you will have to disable it on your device.[19]

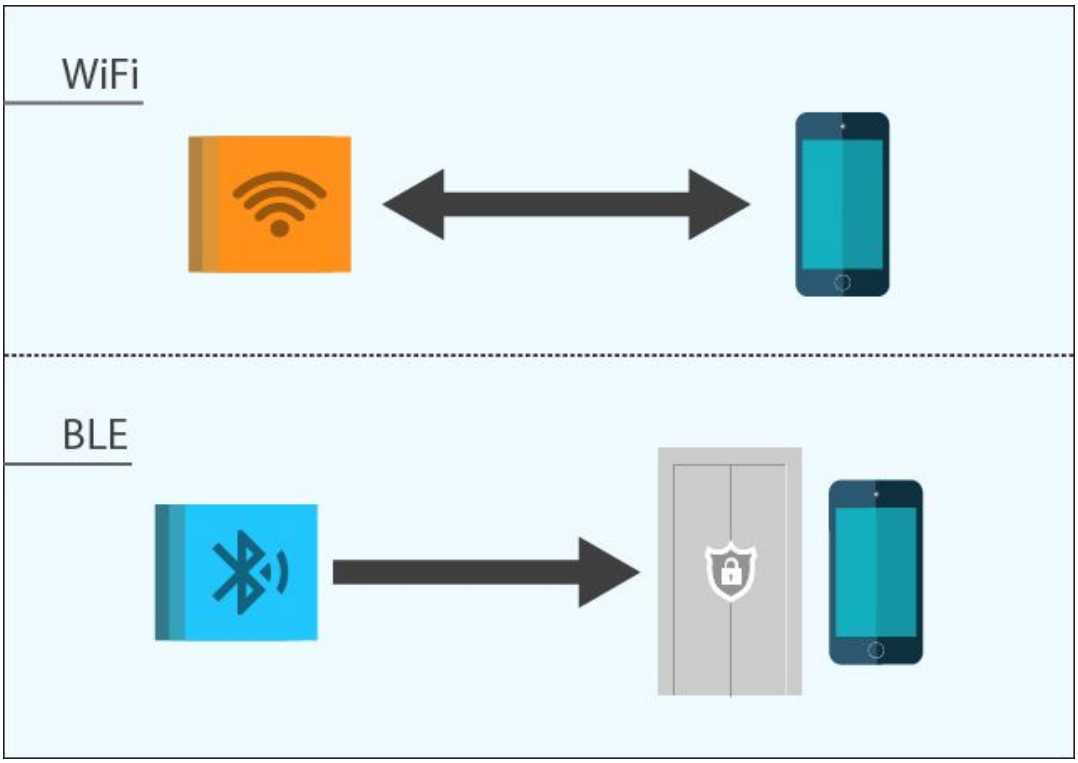

Fig 5.2 : BLE offers more privacy

2. Proximity Detection and Location Detection

Wi-Fi is designed to correctly point to the exact location by measuring access points in a device. Bluetooth is all about proximity, and not exact location. The proximity data provided by BLE is much more accurate than Wi-Fi. When it comes to micro-locating, BLE is the best option, because Wi-Fi signals are really not very capable of penetrating through solid objects, including walls.

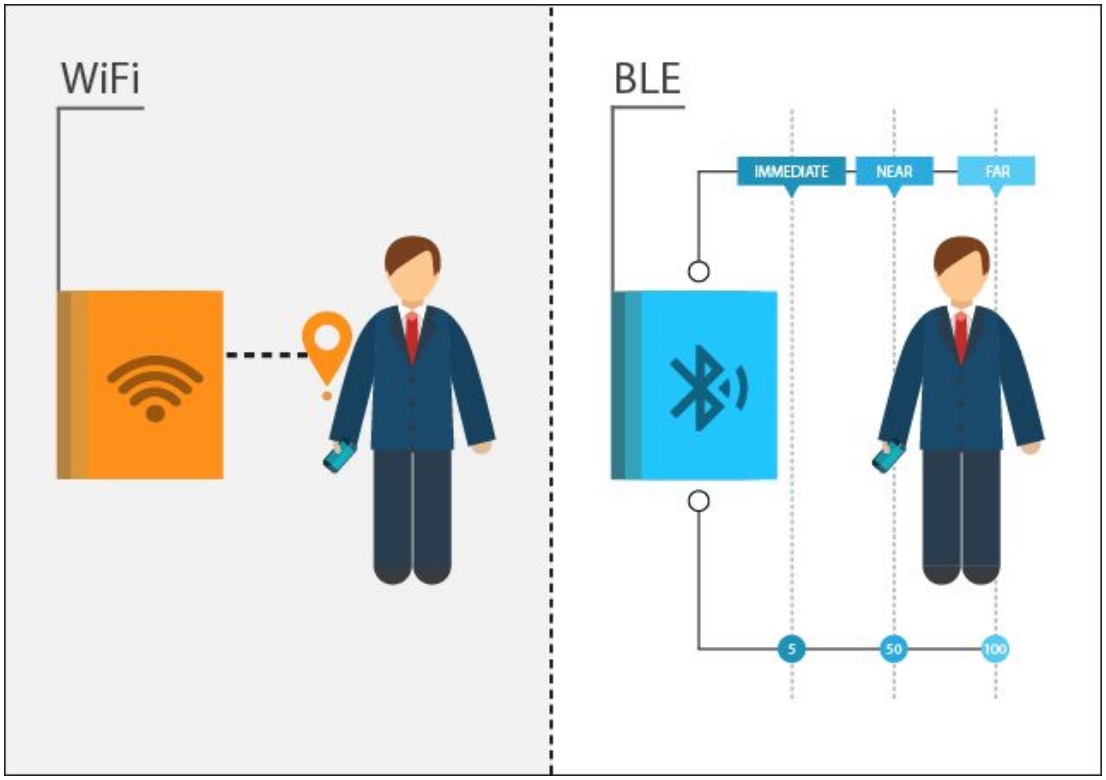

Fig 5.3 : Proximity Detection in BLE

3. Deployment Costs

BLEs are less costly, self-sufficient and can run on a single battery for years, depending on usage. No configuration is required. Wi-Fi needs router configurations, and they have to be connected to a power source. The expense also depends on the router used and of course, the manufacturer.

Finally we talk about the bulkiness of existing systems. The weight of GPS and GSM modules integrated with an arduino or raspberry pi varies from 5g-240g. To eliminate the appended weight, we use the GSM and GPS module on the mobile phone. Our system weighs 33g making it less bulky as compared to existing systems.

CHAPTER VI : CONCLUSION

In the event of an accident, prompt notification and action would minimize the human toll of traumatic injury and death. While semi automatic emergency notification mechanisms are available in some newer vehicles, most automobiles are not equipped with systems that provide automatic notification of data from crash site.

The dedicated accident detection module can be directly fit to vehicles that do not have it when manufactured. The BLE communication technology provides drastic power saving optimizations for the Android device over other similar modules. The possibility of a false alarm is reduced with the help of a user-confirmation switch on the Android application. Emergency services are dispatched in real-time in case of an accident.

The proposed system not only alerts hospitals about the accident but also sends important information like medical history, blood group etc. This allows the hospitals to take action accordingly. Hence, the adoption of an extensive module for both road traffic accident detection and emergency service dispatch helps save crucial time towards post traumatic medical care and helps reduce mortality rates.

FUTURE WORK

We plan to implement a Supervised Machine Learning algorithm to make our system more accurate in detecting incidents and classifying the same as an accident or a false alarm. This will be done by simulating incidents on remote control cars, on which the chip will be attached, and giving the data to the system and telling it what classifies as an accident. This will help the system learn and make it more accurate so as to reduce the rate of false alarms. Also by accumulating data over time we plan on notifying the driver about accident prone areas.

Our aim is to make a subsidised system for a two-wheeler with maximum functionalities. By collaborating with government authorities and bike manufacturers, this project could be associated with a hub. This hub will provide 24*7 customer care service and

also communication with hospital authorities and paramedics. This will provide smooth functioning and will definitely decrease the fatality rate.

Moreover, the app will include a “Parking Mode” which when enabled will notify the user if the vehicle is being towed or being tampered with. The scope of this project includes all requirements gathering, planning, design, development, and implementation of the incident detection device.

CHAPTER VII : REFERENCES

[1] *Road Accidents in India - 2015*. New Delhi: Transport Research Wing, Ministry of Road Transport & Highways, Government of India, 2016, pp. 1-5.

[2] "Low Energy: Point-to-Point | Bluetooth Technology Website", *Bluetooth.com*, 2017. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works/le-p2p.

[3] "Mobile technologies GSM", *ETSI*, 2017. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/mobile/gsm.

[4] "General-purpose input/output", EGR, 2017. [Online]. Available: https://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN_balachandran.pdf

[5] *Road Accidents in India - 2016*. New Delhi: Transport Research Wing, Ministry of Road Transport & Highways, Government of India, 2016, pp. 27-29.

[6] Adnan Bin Faiz, Ahmed Imteaj, Mahfuzulhoq Chowdhury, “Smart Vehicle Accident Detection and Alarming System Using a Smartphone”in *1st International Conference on Computer & Information Engineering*,, Rajshahi, Bangladesh 2016.

[7] Hari Sankar S, Jayadev K, Suraj B and Aparna P, “A Comprehensive Solution To Road Traffic Accident Detection and Ambulance Management”, Department of Electronics and Communication Engineering, NITK, Surathkal, 2016 International Conference on Advances in Electrical, Electronic and System Engineering, 14-16 Nov 2016.

[8] D. Acharya, V. Kumar, N. Garvin, A. Greca and G. Gaddis, "A Sun SPOT based Automatic Vehicular Accident Notification System", in *5th International Conference on Information Technology and Application in Biomedicine,Shenzhen*, China, 2017.

[9] E. Kfoury, E. Nasr and D. Khoury, "An IoT Approach to Vehicle Accident Detection, Reporting, and Navigation", in *Multidisciplinary Conference on Engineering Technology (IMCET)*, Beirut, Lebanon, 2017.

[10] B. Prachi, D. Kasturi and C. Priyanka, "Intelligent Accident-Detection And Ambulance-Rescue System", *International Journal of Scientific & Technology Research*, vol. 3, no. 6, 2014.

[11] A. E. Boualouache, O. Nouali, S. Moussaoui and A. Derder, "A BLE-based data collection system for IoT," *2015 First International Conference on New Technologies of Information and Communication (NTIC)*, Mila, 2015, pp. 2.

[12] B. Energy, "Bluetooth Low Energy | Android Developers", *Developer.android.com*, 2017. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth-le.html. [Accessed: 15- Nov- 2017].

[13] "GATT Overview | Bluetooth Technology Website", *Bluetooth.com*, 2017. [Online]. Available: https://www.bluetooth.com/specifications/gatt/generic-attributes-overview.

[14] "I2C - What's That? - I2C Bus", *I2C Bus*, 2017. [Online]. Available: http://www.i2c-bus.org.

[15] "pysmbus 0.1 : Python Package Index", *Pypi.python.org*, 2017. [Online]. Available: https://pypi.python.org/pypi/pysmbus/0.1.

[16] "CALIBRATION OF SENSORS | Embedded Navigation Solutions", 2017. [Online]. Available: https://www.vectornav.com/support/library/calibration

[17] "GOOGLE MAP APIs | Google Developers", *developers.google.com*, 2017. [Online]. Available: https://developers.google.com/maps/documentation/api-picker

[18] "POWER CONSUMPTION | Raspberry Pi Zero", *lifehacker.com*, 2017. [Online]. Available:https://lifehacker.com/how-much-power-the-raspberry-pi-zero-w-uses-compared-to-1792854782

[19] "BLE v/s WiFi | Comparison of wireless technologies", hackernoon*.com*, 2017. [Online]. Available:https://hackernoon.com/ble-vs-wi-fi-a-comparison-of-wireless-technology-for-iot-product-development-1c7be17f379

SVKM's NMIMS
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Academic Year : 2017-2018 **Course:** B.Tech **Semester:** VII

Project Title : Argus : An Iot based Accident Detection System

Domain : IoT

Mentor Name : PROF. KRISHNA SAMDANI

Type of Project (Internal / External (Industry / NGO / any other)): INTERNAL

Project Team Members

Roll No	Name	Mobile No	Email
E004	Rishabh Nambiar	9619502085	rishabhnambiar.1.nmims@gmail.com
E008	Jeet Parte	9819921027	jeetparte.nmims@gmail.com
E016	Nayanika Shetty	9821127666	shettynayanika@gmail.com
E036	Ayush Sindhwani	7738348585	ayushsindhwani0.nmims@gmail.com
E054	Kartik Prakash	9619648414	kartikprakash.nmims@gmail.com

Rubrics for Synopsis Evaluation (To be filled by Mentor):

Proposal Report (Synopsis)				
Parameter	**Marks out of**	**Accepted (Give Comments)**	**Needs Minor Revision (Give Comments)**	**Needs Major Revision (Give Comments)**
Scope and Limitations	5	(4-5)	(2-3)	(1)
Project Details	10	(8-10)	(4-7)	(Below 4)
Planning & Scheduling (Gantt Chart)	5	(4-5)	(2-3)	(1)

Note : If major revision, attach both synopsis (old and revised)

Note:

- **Marks per week (out of 5)**
- **Total weeks to be considered 12 per semester**